UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESCOBERTA DINÂMICA, SENSÍVEL AO CONTEXTO, DE SERVIÇOS WEB

VICTOR GOMES DA SILVA

ORIENTADOR: PROF. DR. ANTONIO FRANCISCO DO PRADO

São Carlos – SP

Julho/2014

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESCOBERTA DINÂMICA, SENSÍVEL AO CONTEXTO, DE SERVIÇOS WEB

VICTOR GOMES DA SILVA

Dissertação 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, área de concentração: Engenharia de Software

Orientador: Prof. Dr. Antonio Francisco do Prado

São Carlos – SP

Julho/2014

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

S586dd

Silva, Victor Gomes da.
Descoberta dinâmica, sensível ao contexto, de serviços web / Victor Gomes da Silva. -- São Carlos : UFSCar, 2014.
92 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2014.

1. Engenharia de software. 2. Descoberta dinâmica. 3. *Web services*. 4. Computação sensível a contexto. 5. Ontologia. 6. Web semântica. I. Título.

CDD: 005.1 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Descoberta Dinâmica, Sensível ao Contexto, de Serviços Web"

Victor Gomes da Silva

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
(DC/UFSCar)

Prof. Dr. Luis Ferreira Pires
(University of Twente)

São Carlos
Agosto/2014

Dedico este trabalho a Deus que sempre ilumina meus caminhos. A meus familiares, minha querida noiva e a todos os meus amigos

AGRADECIMENTOS

Agradeço a Deus pela minha vida e pelas pedras e tribulações no meu caminho, sem elas não cresceria nem amadureceria para chegar até aqui. Agradeço ao meu pai, João, que sempre me motivou a estudar, ao meu tio Gerson que foi minha inspiração, me incentivando aos estudos e a aprender mais sobre a computação e, claro, minha mãe, **Kikinha**, que esteve sempre ao meu lado, nos momentos alegres e tristes. Aos meus irmãos de sangue, **Gabriel**, **Marina** e **Bruno**, e os de coração da **Rep3ão**, Lú (Graúdo) e Ma, Lê e Bá, Perê e Cléo, Gordinho e Daísa, Mônica e Dieguito, Gabriel e Ana Paula, Rone e Jú, Dani e Amanda, Gustão e Danilão, Beto e Kelly, Emerson e Andressa pelo apoio, consideração e compreensão, por não estar em algumas festas, churras e pagodes por conta da dissertação. Aos meus irmãos de comunidade pelas orações. As minhas tias, que são também conhecidas como mães. A minha noiva, **Ágatha**, que não me deixou desistir dos meus sonhos, esteve sempre ao meu lado me empurrando e me confortando. Aos meus primos, os quais não citarei nomes, senão termino essa dissertação ano que vem. Aos meus amigos de trabalho, pelos conhecimentos técnicos, risadas, pressões, ideias, etc. E a todos que contribuíram e me auxiliaram para realização de mais uma etapa na minha vida.

Agradeço também a todos os funcionários e docentes do Departamento de Computação, principalmente meu orientador Prado, por ter me dado a oportunidade de conhece-lo e trabalhar junto dele. Ele que é uma pessoa muito inteligente, com um nível de conhecimento técnico e pedagógico excelente e que me auxiliou para chegar até aqui.

Não posso me esquecer de todas as pessoas que conheci no PPGCC. A galera mais antiga do LABDES, Dias, Carlão, Baiano, Papotti, Dú, que foi essencial neste trabalho, me auxiliando com os artigos e conhecimentos da área. A galera mais atual, também do LABDES, Mirela, Isis e Tiagão, meu grande parceiro, que me fez dar muitas risadas, com suas piadinhas e saber o quanto é válida a luta pelo tão sonhado título de mestre. Ele que não desistiu de seu sonho e nem poderia, que me deu forças para continuar nessa empreitada. E ainda a galera que está entrando no mestrado Jãum, Guido, Alessandro e Diego Saqui do Doc, que me ajudaram na revisão do meu trabalho e na apresentação também, dando dicas valiosas.

Agradeço de coração meu ex-professor e agora amigo André DT, que sempre me auxiliou e me deu muitas oportunidades que também me possibilitaram estar aqui. Toda a galera do Lapes, Dayr, Thiagão, Elis, Abade, Cleitinho, prof. Sandra, pelas conversas, cafés, pães, chocolates, sem falar nos livros de ENG. Também pelos novos integrantes do PPGCC, Maranhão, Augusto, Luiz, Rener.

Aos meus amigos do PlayKub, Bruninho e Jão. Aos meus amigos do Peladeiros, pelas oportunidades de canetas, chapéis e gols. Toda a rapa de PIS:- Goiás, Fer, Japa (Alan Hiraga), Meninão (Guilherme Stéfano).

Finalmente agradeço a todos que eu me esqueci e que fizeram parte deste momento da minha vida.

Um dia você descobre que se leva muito tempo para se tornar a pessoa que se deseja tornar, e que o tempo é curto. Aprende que não importa até o ponto onde já chegamos, mas para onde estamos, de fato, indo - mas, se você não sabe para onde está indo, qualquer lugar servirá.(...) Aprende que heróis são pessoas que foram suficientemente corajosas para fazer o que era necessário fazer, enfrentando as consequências de seus atos. Aprende que paciência requer muita persistência e prática. Descobre que, algumas vezes, a pessoa que você espera que o chute quando você cai, poderá ser uma das poucas que o ajudará a levantar-se.

William Shakespeare

RESUMO

Atualmente a Internet é um dos maiores meios de comunicação do mundo, transformando e acelerando as formas de publicação e consumo de conteúdo. Ao mesmo tempo, usuários têm se tornado mais exigentes pelo conteúdo e pelo consumo do grande volume de informação que a Web proporciona. Desde a sua criação, a Internet passou por algumas fases, mostrando que as mudanças acontecem de acordo com as exigências dos usuários, como por exemplo, a evolução da Web para Web 2.0 que é marcada pela interação, colaboração e comunicação entre os usuários, estes podendo publicar e compartilhar seus próprios conteúdos na Internet. Esta realidade tem levado à necessidade de construir aplicações que fornecem suporte à grande quantidade de informações para usuários e dispositivos distintos, dispostos em múltiplos contextos de uso, aumentando a demanda por aplicações ubíquas, onde os acessos ocorrem de qualquer lugar, a qualquer momento e a partir de diferentes dispositivos. A construção e manutenção de versões específicas das aplicações para cada contexto de uso tornaram-se difíceis devido à diversidade de dispositivos, usuários, redes de acesso e outros fatores a serem considerados. Além disso, é necessário que as aplicações sejam desenvolvidas mantendo a compatibilidade com as características contextuais do ambiente em que operam, não prejudicando a interação dos usuários. Motivados em vencer estes desafios essa dissertação apresenta uma arquitetura para descoberta dinâmica de serviços sensível ao contexto que apoia a criação de aplicações ubíquas adaptativas a diferentes contextos. As aplicações conduzidas por esta arquitetura atendem a variação de redes de acesso em diferentes contextos. A arquitetura proposta foi baseada nos conceitos da Arquitetura Orientada a Serviços. A arquitetura proposta foi avaliada com um estudo de caso e uma experimentação com usuários. O apoio computacional que a arquitetura proposta emprega traz a vantagem da descoberta de serviços em tempo de execução e que pode variar de acordo com o contexto da rede de acesso.

Palavras-chave: Serviço, Web Service, Busca, Descoberta, Contexto

ABSTRACT

Currently the Internet is one of the biggest media in the world, turning and accelerating forms of publishing and content consumption. At the same time, users have become more demanding for the content and the consumption of large amount of information that the Web provides. Since its inception, the Internet has gone through some phases, showing that the changes take place according to the requirements of users, such as the evolution of the Web to Web 2.0 which is marked by the interaction, collaboration and communication among users, who can publish and share their own content on the Internet. This reality has led to the need to build applications that support the large amount of information to users and different devices, arranged in multiple contexts of use, increasing the demand for software in Ubiquitous Computing, in which access to applications occurs anywhere, anytime and from different devices. The construction and maintenance of specific versions of applications for each context of use have become difficult due to the diversity of devices, users, access networks and other factors to be considered. Moreover, it is necessary for applications to be developed so that the compatibility of the contextual features of the environment in which they operate is maintained and does not impair the interaction of users. Motivated to overcome these challenges this work presents an architecture for Dynamic Discovery of Context-Sensitive Web Services that supports the creation of ubiquitous applications adaptive to different contexts. Applications driven by this architecture meet varying access networks in different contexts. The proposed architecture is based on the concepts of Service Oriented Architecture. The proposed architecture was evaluated with a case study and experimentation with users. The computational support that employs the proposed architecture has the advantage of discovering services at run time and can vary according to the context of the access network.

Keywords: Service, Web Service, Search, Discovery, Context

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Visão Geral

Nesta Seção são apresentadas a contextualização e motivação da pesquisa, assim como os objetivos e estrutura da dissertação.

1.1.1 Contextualização e Motivação

A Web surgiu como um dos maiores meios de comunicação social em todo o mundo, transformando a maneira com que as pessoas comunicam-se e compartilham conteúdo. Desde o seu surgimento, a Web passou por várias fases, de modo que cabe às tecnologias mais recentes atenderem as novas necessidades dos usuários. Com a presença explícita da Internet no cotidiano das pessoas, a utilização de aplicações e serviços em rede tem sido cada vez maior e deverá aumentar nos próximos anos, ao mesmo tempo em que dispositivos móveis com rápida conexão de dados se proliferam rapidamente no mercado (SILVA; PIRES; SINDEREN, 2011). A transição do paradigma tradicional estático para uma Web predominantemente colaborativa e altamente interativa trouxe o conceito da Web 2.0, que foi um marco importante na evolução da Web, fazendo-a explodir em popularidade (FILHO; FERREIRA; VARELLA, 2009).

O uso de aplicações que contém grande volume de informações cresceu rapidamente e, juntamente, a necessidade dos usuários de realizar tarefas complexas e de processar as informações em pouco tempo, criando um desafio aos sistemas computacionais. Diante da miniaturização dos dispositivos computacionais de uso pessoal aliada aos avanços recentes das tecnologias de comunicação sem fio e ao amadurecimento da Computação Ubíqua (WEISER, 1999) ampliaram significativamente as possibilidades de acesso a uma vasta gama de serviços e aplicações de diversos domínios (FORTE; SOUZA; PRADO, 2008). Hoje, por exemplo, é possível ler

e-mails, efetuar transações financeiras, compartilhar recursos (hardware, dados e software) e usufruir de uma série de outras aplicações através de um smartphone, quer o usuário esteja parado ou em movimento, quer esteja em casa, na rua ou no trabalho. Neste sentido tem-se um desafio, o qual visa diminuir a necessidade da interação explícita do usuário com o sistema para obter o que o usuário deseja (VIEIRA; TEDESCO; SALGADO, 2011). Diante desse problema, pesquisadores criaram uma nova abordagem de sistemas, com objetivo de conhecer o contexto em que o usuário se encontra para auxiliá-lo em suas ações. Esses sistemas são chamados de Sistemas Sensíveis ao Contexto (SSC) (VIEIRA; TEDESCO; SALGADO, 2011).

A Web vem ganhando ampla adoção como plataforma de comunicação predominante, com o desafio de criar aplicativos e websites que suportam grandes quantidades de informação de diferentes usuários e dispositivos, que funcionam em diversos contextos de uso. Grande parte do conteúdo disponibilizado na Web é destinada ao consumo humano, e não ao processamento e interpretação por computadores através de processos automatizados (FERREIRA FILHO; VARELLA, 2011). Desta forma, o significado do conteúdo publicado não é considerado em tarefas comuns executadas na Web, como uma busca, por exemplo. Tais atividades acabam sofrendo as consequências da ambiguidade e da baixa relevância. Esse desafio tem atraído o interesse de muitos pesquisadores nos últimos anos. Nesta linha de pesquisa, uma infinidade de soluções foram propostas pela comunidade acadêmica, especialmente com foco em softwares que abordam o desenvolvimento das aplicações sensíveis ao contexto (por exemplo, (DEY, 2000; Dos Santos, 2008; CIRILO et al., 2010)). Este tipo de aplicação refere-se a sistemas de computadores que usam o contexto (isto é, a situação em que o aplicativo é executado) para fornecer serviços ou informações mais relevantes que suportam as tarefas do usuário (DEY, 2000; VIEIRA; CALDAS; SALGADO, 2011). No entanto, a indústria de software tenta resolver o problema da grande variedade de dispositivos de acesso colocando em prática o paradigma da computação orientada a serviços (*Service-Oriented Computing - SOC*), que defende Web Services (WS) como a melhor forma para alavancar a reutilização no desenvolvimento de soluções de software (PAPAZOGLOU, 2003). Para construir o seu modelo de serviço, SOC conta com a Arquitetura Orientada a Serviços (SOA) (ERL, 2009), uma forma de reorganizar as aplicações de software em um conjunto de interação de serviços tecnologicamente neutros, a fim de permitir a integração de baixo acoplamento e uma maior interoperabilidade entre sistemas distribuídos heterogêneos (SILVA et al., 2013).

O modelo arquitetural SOA para WS, conhecido também como arquitetura para WSs, é bem conhecido e baseia-se na interação de três participantes: o *Service Registry*, normalmente referido como *Universal Description, Discovery and Integration (UDDI)*; *Service Provider*; e *Service Requester* (ou *Client*). Este modelo tem uma arquitetura bem definida e estruturada.

Apesar disso, algumas questões ainda permanecem para o desenvolvimento de software baseado em SOA: um dos principais desafios é a realização da descoberta de WS para satisfazer adequadamente as necessidades dos usuários, especialmente em tempo de execução quando o contexto da interação muda constantemente. Mesmo que este modelo já tenha atingido a maturidade, ele está caindo em desuso, uma vez que as novas tecnologias estão ganhando atenção devido à sua facilidade de uso, tais como WSs baseados na arquitetura REST, que são baseados em solicitações HTTP em vez de troca de mensagens tradicional como SOAP. Esta transição tecnológica traz desafios para o desenvolvimento de software baseado em SOA, principalmente em relação a descoberta de serviços e invocação com base em descrições de WSs. Nesta abordagem, são consideradas novas tecnologias de WSs, como WSs baseados em REST e SOAP, para reutilização destes, pois ainda existem muitos WSs baseados em SOAP, e estes precisam ser considerados na descoberta de serviços.

A Web Semântica surge para ajudar a preencher esta lacuna, permitindo a integração entre WS e linguagens semânticas. Utilizando essa tecnologia, surgem os Serviços Web Semânticos (SWS) (BERNERS-LEE; HENDLER; LASSILA, 2001), da mesma forma que serviços web convencionais, eles ficam aguardando a invocação de alguma aplicação através da Internet. Desta forma computadores são capazes de pesquisar, processar, integrar e apresentar o conteúdo destes recursos de forma significativa (FARRAG; SALEH; ALI, 2013).

Cada vez mais as tecnologias de muitos sistemas são integradas para fornecer aos usuários melhores interações. Acompanhando essa tendência, este trabalho apresenta uma abordagem integrada anotando os WSs semanticamente, possibilitando a descoberta dinâmica de WSs sensível ao contexto. Esta abordagem combina a tradicional pilha WS-* (WEERAWARANA et al., 2005) com tecnologias mais atuais, como WSs RESTful e tecnologias da Web Semântica para fornecer aos desenvolvedores uma arquitetura que permite a aplicação executar, em tempo de execução, a descoberta dinâmica de WSs, a fim de promover um comportamento adaptável baseado no contexto de rede. O framework UbiCon (CIRILO et al., 2010) foi estendido com a proposta de fornecer a informação contextual necessária para chegar a sensibilidade de contexto.

1.1.2 Objetivos

Diante dessas motivações e desafios para descoberta dinâmica de serviços em diferentes contextos, o objetivo deste trabalho é **pesquisar e criar uma arquitetura que possibilite a descoberta de serviços web sensível ao contexto**.

Em síntese, o objetivo geral deste trabalho pode ser desmembrado nos seguintes itens:

- Pesquisar trabalhos relacionados a descoberta e composição de serviços;
- Criar uma arquitetura que utilize semântica e contexto como fontes de informação para descoberta dos serviços;
- Reutilizar e estender o framework UbiCon para que dê suporte à descoberta dinâmica de serviços sensível ao contexto da rede, fornecendo atributos de rede que podem culminar em uma melhor execução dos serviços;
- Testar e Avaliar a arquitetura proposta, suportando a publicação e a descoberta de Web Services, possibilitando a reutilização de serviços, considerando as variações contextuais detectadas no ambiente computacional, na rede de acesso a partir do framework UbiCon Estendido.

1.2 Estrutura da Dissertação

Esta Dissertação está organizada em outros cinco capítulos, além deste capítulo introdutório. O conteúdo de cada capítulo é detalhado a seguir.

O **Capítulo 2** apresenta uma revisão da literatura, fundamentos, conceitos e técnicas sobre Padrões utilizados por Web Services e Descoberta de Web Services, Ontologias e Web Semântica assim como Sistemas Sensíveis ao Contexto.

O **Capítulo 3** apresenta a Descoberta de Serviços Sensível ao Contexto. As principais ideias em torno da pesquisa são discutidas e suas atividades são detalhadas, apresentando também a extensão do framework UbiCon.

O **Capítulo 4** aborda a avaliação da proposta, com um estudo de caso e um experimento.

O **Capítulo 5** discute alguns trabalhos encontrados na literatura que relacionam-se ao trabalho apresentado nesta dissertação, apresentando, também, uma comparação entre os mesmos.

O **Capítulo 6** por fim, apresenta algumas conclusões, incluindo as contribuições e limitações deste trabalho, além de possibilidades de trabalhos futuros.

Capítulo 2

FUNDAMENTAÇÃO, CONCEITOS E TÉCNICAS

Este capítulo apresenta os conceitos e técnicas, juntamente com a fundamentação teórica, que relacionam-se ao desenvolvimento deste projeto de mestrado. Na Seção 2.1 são apresentados os conceitos associados a Serviços e Computação Orientada a Serviços. Em seguida, os conceitos referentes as Ontologias e Linguagem Semântica são apresentados na Seção 2.2. Por fim, a Seção 2.3 apresenta os conceitos e a fundamentação sobre Sistemas Sensíveis ao Contexto.

2.1 Serviços e Computação Orientada a Serviços

Serviços estão presentes no dia-a-dia das pessoas, sendo que estas utilizam serviços, instantaneamente, providos por diversas empresas ou até mesmo outras pessoas, como, por exemplo, transporte público, comunicações via Internet ou telefone, fornecimento de energia elétrica, atividades de educação. Todos estes exemplos são serviços prestados por pessoas ou empresas. Serviço é encontrado na literatura para designar vários fenômenos, cada um com diferentes significados dependendo da área que se emprega. Em gestão de organizações, de acordo com Ramaswany (1996), serviço pode ser entendido como as transações de negócios que acontecem entre um provedor (prestador do serviço) e um receptor (cliente) a fim de produzir um resultado que satisfaça o cliente. Esta definição tem um caráter abrangente, pois se aplica em diversas áreas, como na computação, por exemplo. Já para Grönroos (2009) o serviço é uma atividade intangível, que normalmente, mas não necessariamente, acontece durante as interações entre clientes e/ou recursos físicos ou bens e/ou sistemas do fornecedor de serviços, que é fornecida como solução ao problema do cliente. Na área de Tecnologia da Informação (TI), os chamados serviços de TI são definidos como um meio para entregar valor aos clientes (usuários, departamentos, empresas, etc.) através do uso de TI, proporcionando-lhes os resultados desejados a um

baixo custo e com riscos reduzidos (CARTLIDGE et al., 2009). Já na área da Saúde, serviços são ações desempenhadas por profissionais de saúde, com a finalidade de promover, manter ou restaurar a saúde das pessoas (SAÚDE, 2004). Na Figura 2.1, por exemplo, indivíduos prestam serviços diferentes, que juntos, compõem um serviço maior.

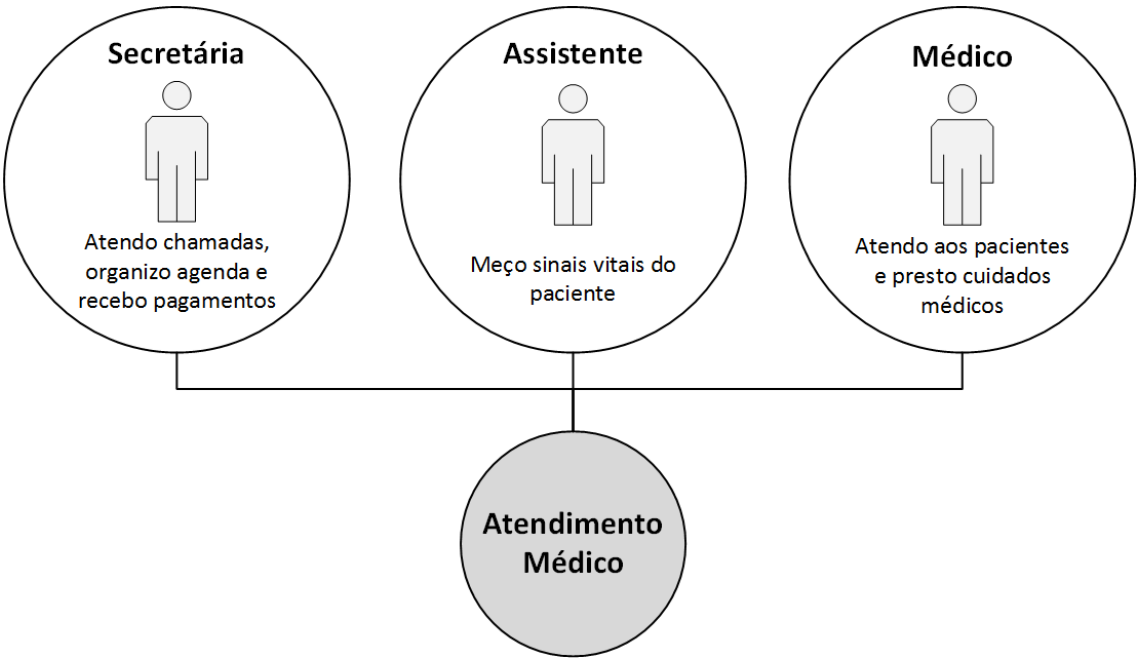


Figura 2.1: Três pessoas, cada uma apta a produzir um serviço diferente. (adaptado de Erl (2009)).

Serviços caracterizam um trabalho realizado ou o fornecimento de algum tipo de capacidade a alguém, a fim de cumprir determinado propósito para satisfação do cliente. Assim, quando uma pessoa ou organização realiza um trabalho em benefício de outro, ou de si mesma, esse trabalho passa a ser considerado um serviço (LUSCH; VARGO, 2006). Os serviços, portanto, são uma forma de entregar algo de valor para alguém.

Pensando no termo serviço em um contexto computacional, é possível comparar as definições citadas anteriormente com diversas interpretações que utilizam o termo serviço nesse contexto. Silva e Lucrédio (2012) fazem a distinção do termo serviço, explicando como utilizar cada tecnologia no seu devido lugar. Seguindo a mesma linha de raciocínio o uso de serviços na computação pode ser implementado por diversas tecnologias, abordadas em diversos paradigmas e arquiteturas. O primeiro conceito que deve ser definido é a Computação Orientada a Serviços (*Service-Oriented Computing - SOC*) (PAPAZOGLOU, 2003), que utiliza o uso de serviços como principal meio de desenvolver aplicações e entregar soluções de software. A SOC representa uma plataforma da computação distribuída, com objetivo de ter serviços dispersos fornecendo capacidades que são invocadas para integrar a solução lógica de diferentes aplicações ou para compor a lógica de outros serviços mais complexos. Os serviços, neste caso,

configuram-se não como uma atividade laboral humana, mas como abstrações em software, que encapsulam uma determinada lógica de programação e reflete um conjunto de capacidades que são úteis em um dado contexto funcional (ERL, 2009).

A Figura 2.2 mostra uma relação entre um serviço prestado no mundo físico e uma abstração do mesmo serviço no contexto da SOC. Este contendo, de forma similar, as mesmas capacidades de um humano.

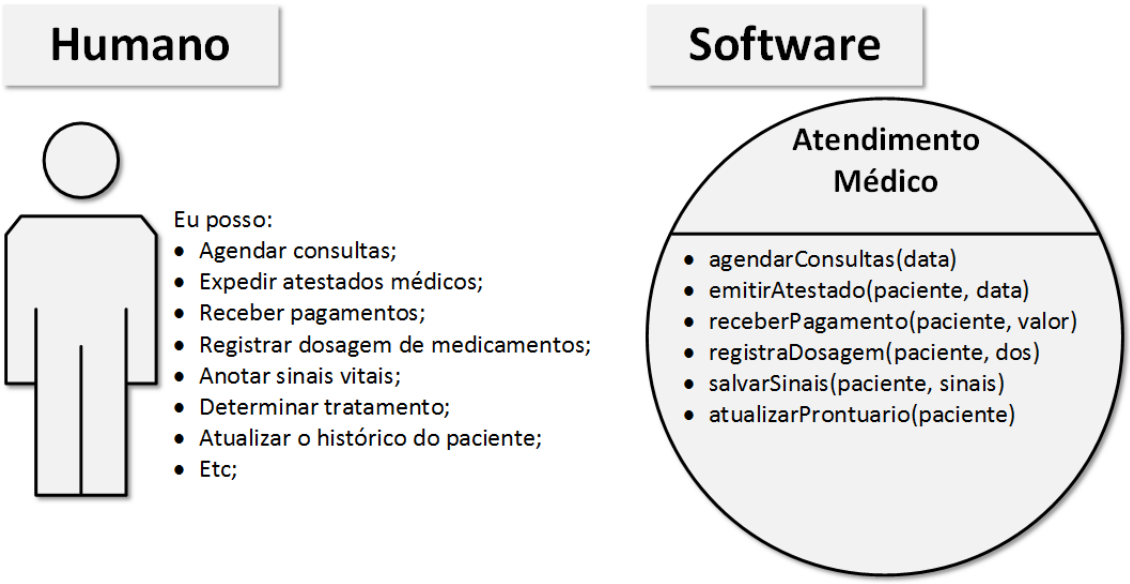


Figura 2.2: Relação entre serviço do mundo físico e abstração do mesmo serviço em software. (adaptado de Erl (2009)).

SOC abrange muitas coisas, incluindo seu próprio paradigma de design e princípios de design, modelo arquitetônico, conceitos e tecnologias relacionadas (ERL, 2009). O modelo arquitetônico utilizado é a Arquitetura Orientada a Serviços (*Service-Oriented Architecture - SOA*). SOA é um modelo arquitetônico que dá suporte e está alinhado aos objetivos da SOC. É, também, uma forma de estruturar as aplicações através de um conjunto de serviços de software, buscando construir funcionalidades de negócios de aplicações com base na composição de funções e recursos de software empacotados como serviços (PAPAZOGLOU, 2003).

Diferentemente de arquiteturas de software convencionais, cujo foco é delinear a estrutura de um sistema de software em subsistemas e componentes interconectados, a SOA foca-se no projeto de sistemas que, por meio de interfaces públicas, usam e fornecem serviços tanto para aplicações de usuários finais como para outros serviços distribuídos em uma rede (PAPAZOGLOU; HEUVEL, 2007). Dentre as finalidades da adoção de SOA como arquitetura estão o aumento da interoperabilidade e a redução do acoplamento entre os componentes básicos de software, buscando, com isso, aprimorar a eficiência, a agilidade e a produtividade no desenvolvimento e manutenção de aplicações em software (ERL, 2009).

Para cumprir com seus objetivos, a SOA determina que os serviços devem ser implementados com interfaces bem definidas, baseadas em padrões abertos e largamente aceitos, e descritos através de uma linguagem padrão de definição de serviços, o que permite que sejam facilmente utilizados para compor as funcionalidades de aplicações. Conforme define a estrutura básica da SOA (Figura 2.3), a interação com os serviços se dá através de troca de mensagens entre agentes de software de clientes e de provedores de serviços (PAPAZOGLOU, 2003).

O provedor é quem mantém o serviço, este podendo ser uma empresa de desenvolvimento ou até um desenvolvedor independente. É o provedor quem fornece e publica o serviço, que é concretizado através de algum agente de software, tornando-o acessível ao cliente, como mostra a Figura 2.3. O cliente, por sua vez, necessita criar uma agente de software que faça a comunicação com o provedor. O cliente pode ser a própria empresa desenvolvedora e mantenedora do serviço ou alguma pessoa ou membro externo que se interesse pelo mesmo.

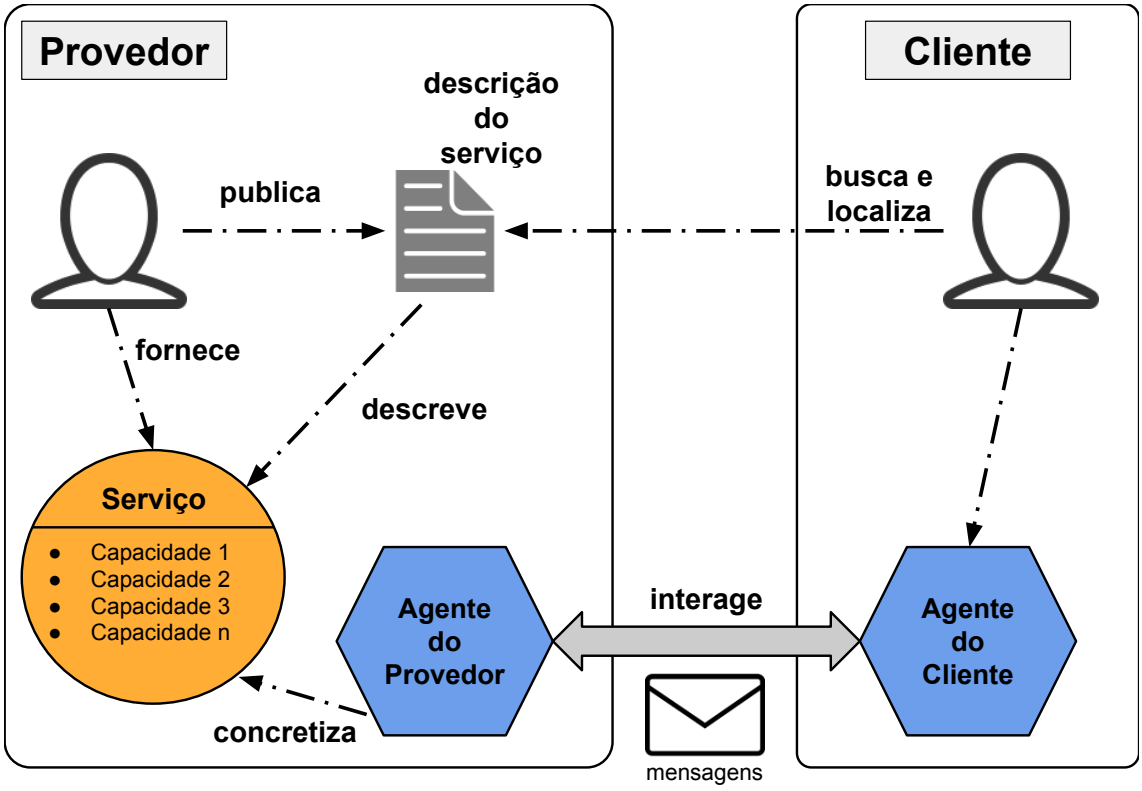


Figura 2.3: Publicação e utilização de serviços por provedor e cliente na SOA

A SOA, muitas vezes tida como uma tecnologia, pode ser confundida com Software como Serviço (*Software as a Service - SaaS*). Segundo Silva e Lucrédio (2012) SaaS é um modelo criado para entregar software na forma de serviço, normalmente acessado por um agente de software do cliente, como o web browser, de forma que o dono do serviço é quem hospeda o software e permite que os usuários executem o serviço sob demanda. SOA e SaaS se complementam e são utilizadas na SOC. Os serviços desenvolvidos e disponibilizados através de

SaaS, com a intenção de reutilização, podem ser agrupados formando um inventário de serviços padronizados (ERL, 2009), que, por sua vez, podem ser utilizados para compor os serviços, atendendo a diversas aplicações tanto dentro como fora da organização provedora.

As capacidades de um serviço são direcionadas para representar funcionalidades de negócio significativas que podem ser combinadas com as de outros serviços para formar diferentes composições de serviços dependendo da necessidade do cliente (PAPAZOGLOU, 2003). Desta forma, as empresas podem intercalar e combinar serviços individuais para desempenharem tarefas específicas de negócio em suas aplicações com um esforço mínimo de programação (PAPAZOGLOU, 2003). Além disso, as organizações podem criar, implantar e integrar múltiplos serviços e coreografar novas funcionalidades de negócio combinando seus ativos de software empacotados como serviços em diversas composições, o que faz da SOA uma alternativa atraente na integração de aplicações antigas e sistemas legados (PAPAZOGLOU; HEUVEL, 2007). O problema destas abordagens é a descoberta dos serviços. Por não ser automatizada, deve haver uma intervenção humana na busca de tais serviços para serem compostos, atrasando o desenvolvimento da composição.

É importante destacar que SOA, SaaS e outras tecnologias estão contidas dentro da SOC. Para facilitar a visualização e diferenciação entre Serviços, SOC, SOA, SaaS, Composição de Serviços e Inventário de Serviços é apresentado na Figura 2.4 um modelo que ilustra a interação destes elementos na SOC.

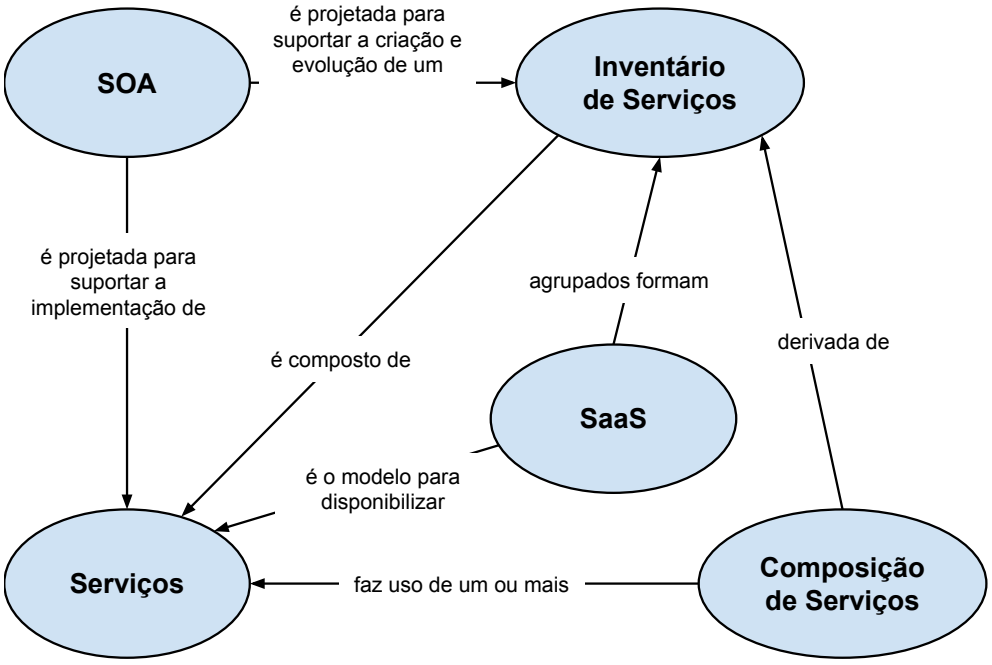


Figura 2.4: Visão conceitual de como os elementos da SOC podem interagir.

Por fim, é possível visualizar a SOA como um modelo arquitetônico que suporta a imple-

mentação de uma variedade de tecnologias. Assim uma empresa que deseja realizar a SOA tem a liberdade de buscar constantemente os objetivos estratégicos associados à computação orientada a serviços, tirando proveito dos futuros avanços da tecnologia. Segundo Erl (2009) e Rosas et al. (2014) no mercado atual, a plataforma de tecnologia mais associada à realização da SOA é a de Web Services, que é explicada na sessão seguinte.

2.1.1 Padrões utilizados por Web Services

É importante diferenciar e posicionar a SOA e os Web Services (WSs). Os WSs surgiram como uma tecnologia que possibilita a realização da SOA, com a intenção de integrar sistemas heterogêneos e disponibilizar aplicações na forma de serviços, com o uso de padrões abertos de Internet, como o HTTP (*Hypertext Transfer Protocol*) e o XML (*eXtensible Markup Language*). O consórcio W3C (Haas; Allen Brown, 2004) define Web Service (WS) como um software projetado para suportar interação máquina-a-máquina em uma rede. Os WSs são independentes da linguagem de implementação, favorecendo a interoperabilidade entre sistemas distintos. Eles utilizam protocolos padrões para envio e troca de mensagens e são identificados através de um Identificador de Recurso Universal (*Uniform Resource Identifier - URI*), o qual é disponibilizado através do descritor de serviço padronizado, juntamente com as assinaturas de seus recursos e funções.

O modelo arquitetural de WSs é baseado na interação de três componentes: Registro (*Registry*), Provedor (*Provider*) e o Cliente (*Requester*), como mostra a Figura 2.5.

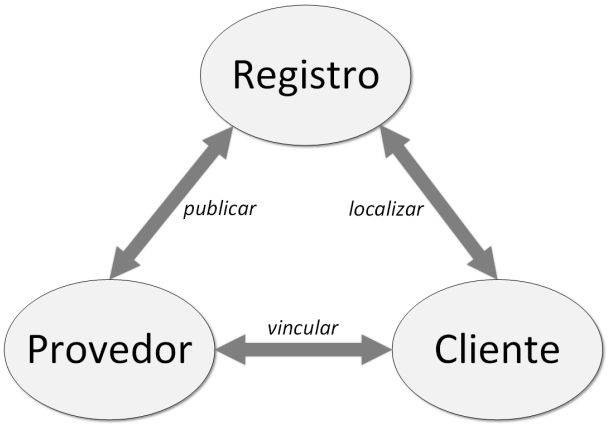


Figura 2.5: Modelo de Web Services

Registro é um inventário de serviços, ou repositório de WSs, onde Provedores de serviços podem publicar as descrições de seus serviços, e Clientes podem encontrar estas descrições. São os Provedores quem fornecem os serviços, que são concretizados em WSs a partir de um software agente do Provedor. O Cliente é responsável por invocar o WS, primeiro solicitando

a descrição de um serviço ao Registro e, logo após, invocando o WS a partir de um software agente do Cliente para comunicação e troca de mensagens com o Provedor.

A interação entre os componentes ocorre através das operações: Publicar (*Publish*), Localizar (*Find*) e Vincular (*Bind*). Quando o Provedor disponibiliza o WS, fornecendo uma descrição que é publicada no Registro acontece uma publicação, usando a operação Publicar. Localizar ocorre quando um Registro é usado pelo Cliente para descobrir e recuperar as informações do serviço, através do descritor do serviço. Por fim, Vincular acontece quando o Cliente usa a descrição do serviço para se vincular com o Provedor e interagir com a implementação do WS. Os WSs podem ser baseados em diferentes tecnologias, embora as mais utilizadas sejam SOAP e REST, que são explicados nas subseções a seguir.

Web Services baseados no protocolo SOAP

O protocolo SOAP (*Simple Object Access Protocol - SOAP*) (Mitra; Lafon, 2007) é um conjunto de convenções definidas pelo consórcio W3C para troca de mensagens SOAP que são transmitidas e negociadas, pela rede, sobre o protocolo HTTP. A troca de informações entre o Cliente e o Provedor do serviço é estruturada, baseada em formato XML e acontece em ambientes descentralizados e distribuídos, com chamadas remotas a procedimentos (*Remote procedure call - RPC*), utilizando o protocolo HTTP como transporte. O SOAP empacota as mensagens a serem trafegadas pela rede em uma estrutura padronizada de envelope, apresentando-se como um mecanismo leve e simplificado para troca de informações estruturadas (Mitra; Lafon, 2007).

Uma mensagem SOAP tem uma estrutura simples, descrita em XML, com um elemento pai, chamado Envelope *Envelope* que é o contêiner que empacota o conteúdo a ser trafegado, e dois filhos, chamados Cabeçalho (*Header*) e Corpo (*Body*) (Figura 2.6). O Cabeçalho é extensível e opcional para mensagem e pode conter informações de QoS (*Quality of service*), configuração e endereçamento. O Corpo é obrigatório e contém a informação útil da mensagem, o que o serviço realmente provê (PAUTASSO; ZIMMERMANN; LEYMANN, 2008). No caso dos WSs a carga útil trata-se das mensagens trocadas com os serviços.

No caso de troca de mensagens através do SOAP, tanto a aplicação servidora, quanto a cliente devem estar aptas a interpretar esse tipo de mensagem, logo, o desenvolvedor deve implementar agentes de software que se comuniquem e entendam esse tipo de protocolo, tendo maior trabalho para desenvolver e manter o sistema. Existem algumas ferramentas que facilitam a implementação, que geram código a partir do descritor de serviço, ou até mesmo do código-fonte, mas ainda sim é um retrabalho gerar e manter esses códigos.

Figura 2.6: Modelo estrutural de um envelope SOAP

Web Services baseados na arquitetura REST

REST é um estilo arquitetural para sistemas hipermídia distribuídos que enfatiza a generalização das interfaces, a escalabilidade da interação entre os componentes e a instalação independente dos mesmos (FIELDING; TAYLOR, 2000). WSs RESTful utilizam a arquitetura REST (*REpresentational State Transfer - REST*) definida por Fielding e Taylor (2000). Este tipo de WS tem características semelhantes ao SOAP, mas em suas diferenças WSs RESTful são mais leves e fáceis de acessar (PAUTASSO; ZIMMERMANN; LEYMANN, 2008; JIANG; LEE; HU, 2012), focam nos recursos do serviço e são transferidos por HTTP, além de usarem os métodos do protocolo HTTP (*GET, PUT, POST, DELETE*) para fazer a comunicação entre Cliente e Provedor, expondo suas capacidades como recursos acessíveis na Web, a qual funciona como uma rede de recursos interligados (FIELDING; TAYLOR, 2000), como ilustra a Figura 2.7.

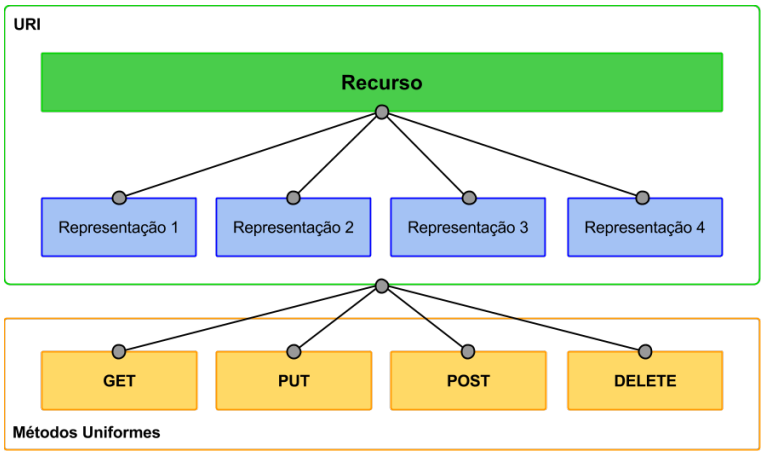


Figura 2.7: Modelo estrutural de um Web Service RESTful

Este tipo de WS está ganhando muita atenção, tanto dos pesquisadores, quanto das empresas (HADLEY, 2009), que estão adotando-o pela facilidade e simplicidade de publicação, invoca-

ção e manutenção (VINOSKI, 2008; BELQASMI et al., 2012; RODRIGUEZ, 2008). Pautasso, Zimmermann e Leymann (2008) afirmam que a implantação e acesso aos WSs RESTful é semelhante a de um website dinâmico.

Apesar do modelo dos WSs estar bem difundido, está caindo em desuso. As empresas fecham acordos off-line e não precisam procurar os WSs em registros de serviço, porque sabem o endereço e, consequentemente, como acessá-los. Ainda existem documentações on-line, como site de desenvolvedores, que publicam como acessar seus serviços evitando o uso de descritores, o que faz com que seja desnecessário o uso do Registro de serviços, não aderindo ao modelo citado (PAUTASSO; ZIMMERMANN; LEYMANN, 2008; JIANG; LEE; HU, 2012; BELQASMI; GLITHO; FU, 2011; SHENG et al., 2014).

Descritores de Web Services

O modelo de WSs provê meios de comunicação entre suas entidades, trocando informações através de mensagens e descritores de WSs. Os descritores descrevem, sintaticamente, como acessar o serviço e o que o serviço provê. O consórcio W3C padronizou o formato do descritor, nomeando-o WSDL (*Web Service Description Language*), para descrever os WSs de uma forma estruturada. Segundo o W3C (Curbera; Weerawarana, 2001) o WSDL define uma gramática e um modelo XML para descrever serviços de rede como coleções de pontos finais de comunicação capazes de trocar mensagens. Uma descrição de WS é um documento pelo qual o Provedor comunica as especificações para o Cliente invocar o serviço, definindo, assim, como deve ser a interação entre eles, como e onde deve ser o acesso e quais são os dados de entrada e saída que ele aceita.

A primeira versão da WSDL surgiu em meados do ano 2001 e suportava descrever somente WSs baseados em SOAP, não prevendo a descrição de WSs com comunicações puramente em HTTP (Moreau; Weerawarana, 2007), como ocorre com os que seguem os princípios do REST. Logo após, com o surgimento dos WSs baseados em REST, que utilizam grande parte das características do protocolo HTTP, para troca de mensagens entre Provedor e Cliente, o W3C lançou uma atualização da versão da WSDL para suportar também a descrição de WSs RESTful, chamada de *WSDL 2.0*. A WSDL descreve os WSs em dois estágios principais: abstrato e concreto (Figura 2.8). No abstrato a WSDL descreve um serviço em termos das mensagens que envia e recebe, já no concreto, uma ligação especifica detalhes de transporte e de formato para uma ou mais interfaces. A WSDL é dividida em 4 partes: Types, Interfaces, Binding e Service, sendo que Types e Interfaces são do estágio abstrato e Binding e Service são do estágio concreto. Types determina o tipo de mensagens o serviço enviará e que tipo de

mensagem pode receber e processar. Interfaces detalha o que o serviço provê. Binding detalha como acessar o serviço. Service detalha onde o serviço pode ser acessado, mostra o endereço, o ponto final (Moreau; Weerawarana, 2007).

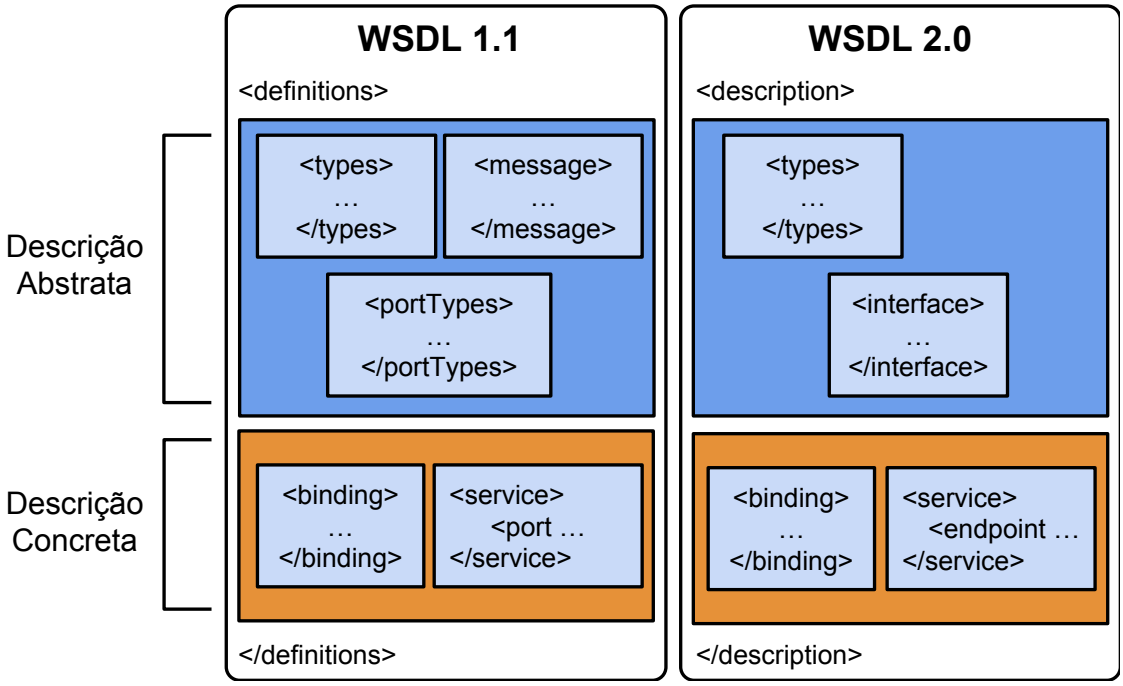


Figura 2.8: Diferença entre as versões do WSDL

Além do WSDL, que é o descritor de WSs padrão, existe ainda o WADL, proposto por Hadley, para prover uma descrição processável por máquina, de recursos RESTFul, que são baseados em HTTP (HADLEY, 2009). Um grande número de empresas baseadas na Web (Google, Yahoo, Amazon e outras) tem utilizado RESTful (HADLEY, 2009) para prover acesso aos seus dados internos, mas utilizam documentação com web sites, ao invés de usarem o WADL. WADL se diferencia do WSDL pois provê uma interface, em XML, descrevendo uma Aplicação e não um Serviço, mapeando os conceitos que formam o paradigma RESTful (FILHO; FERREIRA; VARELLA, 2009). O WADL contém o elemento Root Application que engloba os elementos Grammars e Resources. As Grammars para definição do formato de dados que serão trocados e o Resources contém os recursos fornecidos pela aplicação. Dentro de Resources pode conter vários recursos que são definidos através de 3 elementos: Doc, Method, Resource. O Doc provê um documento sobre o recurso, que pode ser o próprio WADL. O Method apresenta um dos métodos do protocolo HTTP que é utilizado para invocar o recurso e descreve as entradas e saídas do recurso. O Resource descreve sub-recursos e mostra o caminho para acessar o recurso da aplicação.

Apesar do WADL descrever de forma mais fácil e eficiente os WSs RESTful, o WSDL descreve tanto WSs baseados em SOAP quanto em RESTful, o que facilita a descoberta e in-

vocação de ambos tipos de serviço, justificando o uso do padrão WSDL. Por fim, é interessante ressaltar que uma desvantagem na descoberta dinâmica de WSs é que os descritores englobam apenas descrição sintática de seus serviços, se fazendo necessário uma descrição semântica para obtenção de melhores resultados nas buscas.

UDDI

Quando os WSs são construídos, eles devem ser acessados por alguma aplicação cliente, em algum lugar na web, seja por um buscador dinâmico ou de modo estático. Uma forma de acessar um WS é fazer a aplicação cliente conhecer o endereço do serviço, desta maneira caracterizando o modo estático de localização e acesso a um serviço, sendo assim, a aplicação cliente necessita saber o endereço do WS para acessá-lo e invocá-lo. Porém, ele pode ser descoberto, caracterizando o modo dinâmico de descoberta de um WS. UDDI (*Universal Description Discovery and Integration*) é uma estrutura independente de plataforma feita para descrever e encontrar serviços, previamente publicados, dinamicamente, que está contida no modelo dos WSs. Segundo dados da OASIS (OASIS, 2002), que mantém as especificações do UDDI, estas especificações definem um serviço de registro de WSs. Um serviço de registro, conhecido como UDDI Registry, é quem gerencia as informações sobre prestadoras de serviços, implementações de serviços e seus dados. Os Provedores podem usar o UDDI para anunciar seus serviços e os Clientes podem utilizá-lo para descobrir serviços que atendam às suas necessidades e invocá-los. Logo, temos que UDDI refere-se a um repositório de WSs, providos por empresas e seus descritores.

Dustdar e Schreiner (2005) e Sheng et al. (2014) afirmam que UDDI e outros modelos de registro de dados baseados no UDDI foram implementados, mas não são amplamente utilizados, e em caso de descoberta dinâmica de serviço, ele ainda não cumpre os requisitos. O UDDI faz a busca dos serviços armazenados por palavras-chave, dificultando a descoberta dinâmica dos serviços. Por exemplo, o usuário busca um serviço a palavra 'Login', mas o serviço possui a palavra-chave 'Log-in' vinculada a ele. Para resolver esse problema, pesquisas giram em torno de descoberta dinâmica através da semântica, baseada em ontologias. A descoberta semântica proporciona a busca através de conteúdo e não somente por palavras-chave, facilitando o entendimento pelo computador (MARTIN et al., 2004). A descoberta de um WS no UDDI depende do descritor do serviço, nesse caso, o WSDL. A WSDL complementa o padrão UDDI, proporcionando um modo uniforme de descrever a interface e a ligação entre Provedor e Cliente. Como as especificações do UDDI não definem um tratamento semântico para os WSs, o UDDI deve ser estendido para compreender o contexto semântico dos serviços, favorecendo, a

descoberta dinâmica de serviços semânticos dentro do UDDI.

2.1.2 Descoberta de Web Services

Durante essa pesquisa, dois tipos de descoberta de serviços foram identificadas. Estratégias de descoberta estática e dinâmica, que dizem respeito ao momento em que os serviços são descobertos. Elas são equivalentes ao tempo de projeto e tempo de execução (DUSTDAR; SCHREINER, 2005; SHENG et al., 2014) respectivamente. A descoberta estática ocorre durante o tempo de projeto, quando a arquitetura e o projeto do sistema são planejados. Os componentes a serem utilizados são escolhidos, ligados entre si, e, finalmente, compilados e implantados. Isso pode funcionar muito bem, desde que o ambiente de serviço (parceiros de negócios e componentes de serviço) não sofram mudanças frequentemente. Descoberta dinâmica ocorre em tempo de execução, em que a aplicação já está sendo executada. Esses dois tipos são os mais utilizados quando menciona-se descoberta de serviços.

2.2 Ontologias e Web Semântica

Grande parte do conteúdo disponibilizado na Web é destinada ao consumo humano, e não ao processamento e interpretação por computadores através de processos automatizados (FERREIRA FILHO; VARELLA, 2011). Desta forma, o significado do conteúdo publicado não é considerado em tarefas comuns executadas na Web, como uma busca, por exemplo. Tais atividades acabam sofrendo as consequências da ambiguidade e da baixa relevância.

Proposta por Berners-Lee, Hendler e Lassila (2001), a Web Semântica surge com objetivo de auxiliar o conteúdo publicado na Web através de anotações semânticas formais, o que minimiza as limitações da Web, permitindo a manipulação do conteúdo por aplicações com capacidade para interpretar informações semânticas (FORTE; SOUZA; PRADO, 2008). Neste contexto, sintaxe e semântica passam a compartilhar o mesmo grau de importância nos processos de publicação, busca e invocação de informações (FERREIRA FILHO; VARELLA, 2011).

A construção de ontologias, necessariamente, requer uma linguagem apropriada. Esta linguagem deve ter uma boa semântica definida, ser suficientemente expressiva para descrever relações complexas entre objetos e ser capaz de manipulação automática e inferência, tudo dentro dos limites aceitáveis de tempo e de recursos (MCILRAITH; MARTIN, 2003).

O W3C guia o desenvolvimento, organização e padronização de linguagens para promover interoperabilidade entre aplicações web (FORTE; SOUZA; PRADO, 2008). Essas linguagens

incluem RDF (*Resource Description Framework - RDF*) (MANOLA; MILLER; MCBRIDE, 2004) e, mais recentemente, a OWL (*Ontology Web Language - OWL*)(MCGUINNESS; HARMELEN et al., 2004), que são linguagens de representação do conhecimento.

2.2.1 Ontologia

A literatura sobre ontologias apresenta uma série de definições distintas, contendo variações no significado de acordo com o contexto em que se utiliza. O termo e o conceito *Ontologia* foi introduzido pelo filósofo grego Aristóteles (384 - 322 a.C.) com uma teoria sobre a natureza da existência, preocupando-se em prover um sistema de categorização do conhecimento.

Gruber et al. (1995) define uma ontologia como uma especificação formal e explícita de uma conceitualização compartilhada. De forma mais genérica Hendler (2001) traz a definição de ontologia como sendo um conjunto de termos do conhecimento, incluindo o vocabulário, as interconexões semânticas e algumas regras simples de inferência e lógica para algum tópico em particular.

As ontologias têm um papel muito importante na automatização da descoberta, composição e adaptação de Web Services. São elas que descrevem e dão conhecimento para que o computador possa processar e entender o conteúdo dos Web Services, podendo assim, descobrir, compor, invocar e adaptar os serviços disponíveis, automaticamente. A definição mais utilizada de ontologia na literatura, sobre serviços de web semântica é: Ontologia é um conjunto de serviços que compartilham o mesmo domínio de interesse e descrevem como serviços web podem ser descritos e acessados (DUSTDAR; SCHREINER, 2005; SHENG et al., 2014). Desta maneira, Ontologias podem aprimorar o funcionamento da Web de diversas formas. Elas podem ser utilizadas para melhorar a precisão dos mecanismos de busca, ou ainda relacionar o conteúdo de uma página Web ao conhecimento real sobre os dados nela contidos (BERNERS-LEE; HENDLER; LASSILA, 2001).

O conhecimento processável pelo computador se dá através das ontologias. As ontologias formalizadas através de expressões semânticas, que expressam o conhecimento em termos computacionais. Essas descrições são criadas a partir de linguagens de representação do conhecimento vista nas subseções 2.2.2 e 2.2.3.

2.2.2 OWL

OWL é uma linguagem de marcação utilizada na publicação e compartilhamento de ontologias na Web. Nesta linguagem, ontologia é um conjunto de definições de classes, características

de propriedades e restrições de cardinalidade, que relacionam o modo que estas classes, características e propriedades podem ser utilizadas.

A OWL é projetada para aplicações que necessitam processar o conteúdo da informação, proporcionando uma maior interpretabilidade do conteúdo da Web pelos computadores do que outras tecnologias, como, por exemplo, XML, RDF e RDF *Schema*, disponibilizando um vocabulário adicional com uma semântica formal (HERMAN, 2004). Entre estas linguagens a OWL vem se destacando no meio acadêmico e segundo Isotani et al. (2009), o sucesso da OWL fez com que ela se tornasse o padrão utilizado pela comunidade de Web Semântica, especialmente para descrever ontologias.

A linguagem OWL oferece três sublinguagens projetadas para uso específico de diferentes tipos de desenvolvedores (VIEIRA et al., 2005; HERMAN, 2004; FILHO; FERREIRA; VARELLA, 2009).

OWL Lite suporta usuários que necessitam principalmente de uma hierarquia de classificação e as características de restrição simples. Por exemplo, enquanto OWL Lite suporta restrições de cardinalidade, ela só permite valores de cardinalidade 0 ou 1. É mais simples fornecer suporte a ferramenta para OWL Lite que seus parentes mais expressivos, e fornecer um caminho de migração rápido para outras taxonomias.

OWL DL suporta a máxima expressividade sem perder completude computacional (todas as vinculações são garantidas para ser computar os dados) e decidibilidade (todas as computações terminarão em tempo finito) de sistemas de raciocínio. OWL DL inclui toda a linguagem OWL com restrições, como o tipo de separação (uma classe não pode ser um indivíduo e uma propriedade ao mesmo tempo, a propriedade não pode ser um indivíduo ou classe ao mesmo tempo). OWL DL é assim chamado devido à sua correspondência com lógica descritiva, um campo de pesquisa que estuda um fragmento de decisão particular de lógica de primeira ordem. OWL DL foi concebido para apoiar o segmento de negócio existente e tem propriedades computacionais desejáveis para sistemas de raciocínio.

OWL Full é destinado a desenvolvedores que querem máxima expressividade e a liberdade sintática do RDF sem garantias computacionais. Por exemplo, em OWL Full, uma classe pode ser tratada simultaneamente como uma coleção de indivíduos e como um indivíduo em seu próprio direito. Outra diferença significativa de OWL DL é que uma propriedade DatatypeProperty pode ser marcado como uma propriedade InverseFunctionalProperty. OWL Full permite uma ontologia aumentar o significado do (RDF ou OWL) vocabulário pré-definido. É improvável que qualquer software raciocínio será capaz de suportar todas

as funcionalidades do OWL Full.

Cada um desses sublinguagens é uma extensão do seu antecessor mais simples.

2.2.3 OWL-S

Sabendo que a descrição semântica é essencial para descoberta eficiente de Web Services, pesquisas tem mostrado que a ontologia OWL-S, a qual estendeu a OWL criando uma ontologia para serviços, é mais eficiente no contexto de descrição semântica para Web Services (FORTE; SOUZA; PRADO, 2008). A OWL-S é uma ontologia para Web Services que permite descobrir, compor, invocar e monitorar serviços com um auto grau de automação.

A OWL-S é baseada em classes. Ela é composta por 1 classe principal (*Service*) e outras 3 classes que derivam desta (Figura 2.9), que são (MARTIN et al., 2004):

- *Service Profile*: A classe Service Profile diz "o que o serviço faz", de uma maneira apropriada para um Cliente, determinando se o serviço corresponde às suas necessidades. Descreve o que é realizado pelo serviço, suas limitações de aplicabilidade e qualidade, e as exigências que o solicitante do serviço deve satisfazer para usá-lo com sucesso;
- *Service Model* - A classe Service Model diz ao Cliente como utilizar o serviço. Detalha o conteúdo semântico dos pedidos, as condições em que os resultados particulares irão ocorrer, e, se necessário, o passo a passo dos processos que levam a estes resultados. Ou seja, ele descreve como invocar o serviço e o que acontece quando o serviço é realizado.
- *Service Grounding* - A classe Service Grounding especifica os detalhes de como o Cliente pode acessar um serviço. Especifica um protocolo de comunicação, formatos de mensagem, e outros serviços específicos, por exemplo, detalhes de números de porta utilizados em contato com o serviço. Além disso, deve especificar, para cada tipo de semântica de entrada ou de saída especificado na Service Model, uma maneira, de forma não ambígua, para troca de dados com o serviço (isto é, fornece detalhes sobre protocolos de comunicação e formato de mensagens).

Uma vez que as descrições de serviços OWL-S são baseados em documentos OWL, todas as características do modelo de domínio OWL podem ser utilizadas para estruturar as descrições de serviço, facilitando a reutilização de ontologias OWL anteriormente desenvolvidas (FORTE; SOUZA; PRADO, 2008).

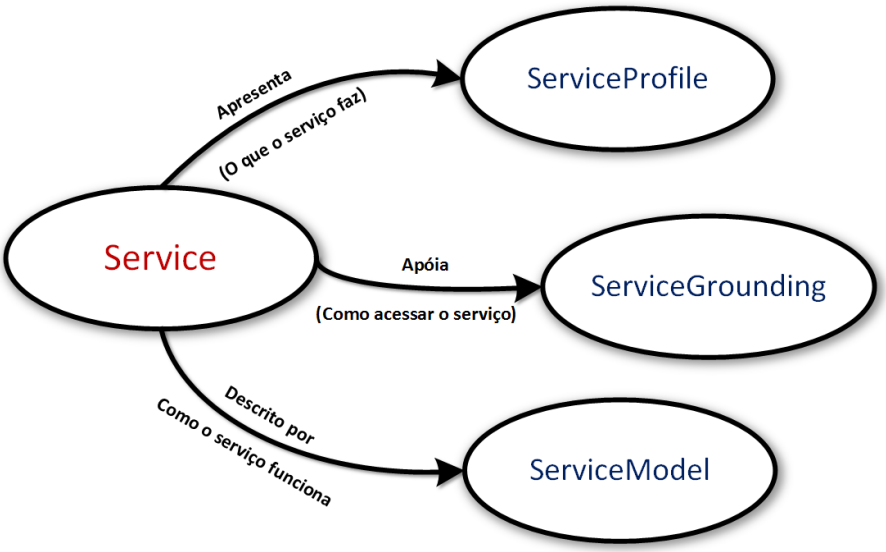


Figura 2.9: Estrutura da Ontologia utilizada para descrever Web Service. Adaptado de Martin et al. (2004).

2.2.4 Web Services Semânticos

Unindo serviços com linguagem semântica temos serviços semânticos, os quais lidam com as limitações existentes nos atuais WSs através do aprimoramento da descrição dos serviços, definindo uma camada semântica com o objetivo de alcançar processos automáticos de descoberta, composição, monitoramento e execução (ANTONIOU; Van Harmelen, 2004). Para alcançar esse objetivo cientistas têm usado 4 atributos descritos através da semântica, na classe *Service Profile* da OWL-S: *Inputs, Outputs, Preconditions and Effects - IOPE* (FORTE; SOUZA; PRADO, 2008; BELLUR; VADODARIA, 2009; REDDY; DAMODARAM, 2012; SANTOS; WIVES; DE OLIVEIRA, 2012). Certamente existem funções que são exclusivas da OWL-S, e a principal delas é o mapeamento entre as especificações OWL-S e as especificações sintáticas do WSDL, o qual, permite a integração das ontologias com os WSs. Este mapeamento é de um descritor para uma ontologia. Além disso, a Service Profile importa as definições semânticas definidas em outros Service Profiles e em outras ontologias, facilitando o reuso e evitando ambiguidades (FORTE; SOUZA; PRADO, 2008).

Finalmente temos Web Services Semânticos que podem ser buscados por conteúdos e não apenas por palavras-chave, aumentando o poder da descoberta de WSs em Registros.

2.3 Sistemas Sensíveis ao Contexto (SSC)

Com a presença explícita da Internet no cotidiano das pessoas, o uso de aplicações que contém grande volume de informações está crescendo rapidamente e, juntamente, a necessidade dos usuários de realizar tarefas complexas e de processar as informações em pouco tempo, criando, assim, um desafio aos sistemas computacionais. O desafio visa diminuir a necessidade da interação explícita do usuário com o sistema para obter o que o usuário deseja (VIEIRA; TEDESCO; SALGADO, 2011). Diante desse problema, pesquisadores criaram uma nova abordagem de sistemas, que tem por objetivo conhecer o contexto em que o usuário se encontra para auxiliá-lo em suas ações. Esses sistemas são chamados de Sistemas Sensíveis ao Contexto (VIEIRA; TEDESCO; SALGADO, 2011) ou Sistemas Cientes de Contexto (DEY, 2000). SSCs são sistemas de computador que usam o contexto para fornecer serviços ou informações mais relevantes dando suporte as tarefas do usuário, onde estas tarefas dependem do contexto (VIEIRA; TEDESCO; SALGADO, 2011; DEY, 2000). Este trabalho refere-se ao termo sistemas sensíveis ao contexto pelo fato de traduzir melhor a semântica de um sistema que percebe mudanças no contexto, adapta-se e reage a estas mudanças (Dos Santos, 2008).

2.3.1 Contexto

O Contexto é a peça fundamental para filtrar as informações disponíveis e transformá-las em informações relevantes. Na literatura encontram-se diversas definições de contexto. Segundo Vieira (VIEIRA; TEDESCO; SALGADO, 2011) contexto é tudo que envolve uma situação, em um dado momento, e que permite identificar o que é ou não relevante para interpretar e compreender a situação. Dos Santos (2008) ainda faz a distinção de dois conceitos: Contexto e Elemento Contextual (EC), alegando que EC é qualquer dado, informação ou conhecimento que permite caracterizar uma entidade em um domínio. Conforme definido em (DEY, 2000), o contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade (por exemplo, pessoa, lugar, objeto, usuário do aplicativo). Contexto envolve informações que se referem a vários aspectos associados ao funcionamento da aplicação, tais como usuário, o dispositivo de acesso, o ambiente, a rede, dentre outros (DEY, 2001). Utilizando estas ideias, Contexto, em sistemas computacionais, é um instrumento de apoio à comunicação entre os sistemas e seus usuários. A partir da compreensão do contexto, o sistema pode, em circunstâncias diversas, mudar sua sequência de ações, o estilo das interações e o tipo de informação fornecida aos usuários de modo a adaptar-se às necessidades atuais destes. Sistemas que utilizam contexto para direcionar ações e comportamentos recebem o nome de Sistemas Sensíveis ao Contexto ou Cientes de Contexto (Dos Santos, 2008).

2.3.2 Aplicações de SSC

Há uma grande diferença entre aplicações tradicionais e Aplicações Sensíveis ao Contexto (ASCs). As tradicionais agem levando em consideração somente as solicitações e informações fornecidas explicitamente pelos usuários. Já as aplicações sensíveis ao contexto, diferem destas principalmente no total de dados de entrada e saída. Os dados de entrada podem ser informações explícitas fornecidas pelo usuário, armazenadas em bases de conhecimento contextuais, inferidas por meio de raciocínio e, ainda, aquelas percebidas a partir do monitoramento do ambiente (Dos Santos, 2008). Os dados de saída podem variar de acordo com o contexto e com a solicitação do usuário.

As funcionalidades básicas para construção de uma aplicação sensível ao contexto são listadas por Vieira (Dos Santos, 2008), definindo-as em 3 categorias:

- Especificação do Contexto: cuida do levantamento dos requisitos de contexto e da modelagem das informações contextuais necessárias para a aplicação.
- Gerenciamento do Contexto: indica como o contexto será manipulado pelo sistema, em termos das seguintes tarefas: aquisição, armazenamento, processamento e disseminação de Elementos Contextuais.
- Uso do Contexto: define como o contexto influencia o comportamento do sistema e como será efetivamente utilizado.

Tendo em vista que as informações do ambiente podem ser consideradas contexto deve-se que extrair as informações relevantes. Dos Santos (2008) em sua tese define um framework chamado CEManTIKA (*Contextual Elements Modeling and Management through Incremental Knowledge Acquisition*) para obtenção, gerenciamento e disseminação do contexto computacional.

Para melhor ilustrar o funcionamento das aplicações sensíveis ao contexto considere, por exemplo, o caso de uma aplicação para auxílio à marcação de consultas médicas que leva em consideração as preferências dos pacientes e seu contexto. O Paciente cadastra seu dados com suas preferências na base de dados compartilhada por clínicas e hospitais. Quando o Paciente procurar por um médico para agendar uma consulta, a aplicação age ajudando-o em suas ações. O Paciente pode informar sua enfermidade ou a especialidade médica requerida para filtrar a busca por clínicas e centros adequados. O sistema localiza clínicas apropriadas próximas ao paciente, considerando sua localização e preferências, e busca por médicos com as especialidades requeridas, levando em consideração a agenda do médico para marcar a consulta. O

sistema emite mensagens de alerta tanto para o paciente quanto para o médico caso a consulta seja agendada com sucesso, ou na ocorrência de alguma falha.

2.3.3 Arquitetura de SSC

Aplicações sensíveis ao contexto podem ser implementadas de diversas maneiras. A arquitetura da aplicação depende de requisitos e condições especiais, dentre os quais o método de aquisição dos elementos contextuais exerce influência considerável na definição do estilo arquitetural das ASCs (DUSTDAR; SCHREINER, 2005; SHENG et al., 2014). Perera et al. (2013) realizaram uma grande pesquisa sobre arquiteturas de SSC, mostrando os pontos fortes e fracos de cada arquitetura, passando por todas as fases do gerenciamento de contexto. Embora não tenha comparado com a abordagem de Dos Santos (2008) a solução proposta por Perera et al. (2013) em seu estudo é muito similar a de Dos Santos (2008), a qual é utilizada neste projeto de pesquisa. Existem três diferentes abordagens para aquisição de ECs, listadas por Chen, Finin e Joshi (2003):

- Acesso direto às fontes de contexto: nesta abordagem, a aplicação cliente obtém os elementos contextuais diretamente das fontes de contexto disponíveis. Não há camadas para tratamento do contexto, de forma que o código para aquisição e processamento dos elementos contextuais fique atrelado ao código das regras de negócio da aplicação;

- Infraestrutura intermediária: esta abordagem incorpora uma arquitetura intermediária às ASCs que encapsula a manipulação do contexto, ocultando os detalhes de baixo nível de obtenção dos elementos contextuais de suas fontes de origem. Esta abordagem favorece a extensibilidade, uma vez que o código da aplicação cliente não necessita ser modificado caso haja alterações na forma de aquisição dos ECs;

- Servidor de contexto: esta abordagem distribuída estende a arquitetura baseada em infraestrutura intermediária, introduzindo um componente remoto de gerenciamento de acesso às fontes de contexto. A obtenção dos ECs é realizada em um servidor, chamado servidor de contexto, para facilitar múltiplos acessos simultâneos. Esta abordagem alivia as aplicações clientes das operações intensivas para obtenção dos ECs de suas fontes de origem, o que é um aspecto importante considerando uma ASC que executa em dispositivos com recursos limitados.

Entre estas abordagens, o uso da infraestrutura intermediária para gerenciar o contexto introduz uma arquitetura em camadas à ASC. Isso possibilita ocultar os detalhes de baixo nível

para extração dos ECs de suas fontes de contexto originais em uma camada projetada especificamente para gerenciar o contexto. Ao contrário da abordagem de acesso direto às fontes de contexto, a infraestrutura intermediária, além de encapsular as funcionalidades relacionadas à obtenção do contexto favorecendo a extensibilidade, também possibilita o reuso da camada de gerenciamento de contexto em diferentes aplicações (BALDAUF; DUSTDAR; ROSENBERG, 2007), uma vez que essa camada pode ser considerada como independente de domínio (Dos Santos, 2008).

A Figura 2.10 apresenta um modelo conceitual de arquitetura em camadas para ASCs, chamada Arquitetura de Contexto (Dos Santos, 2008)).

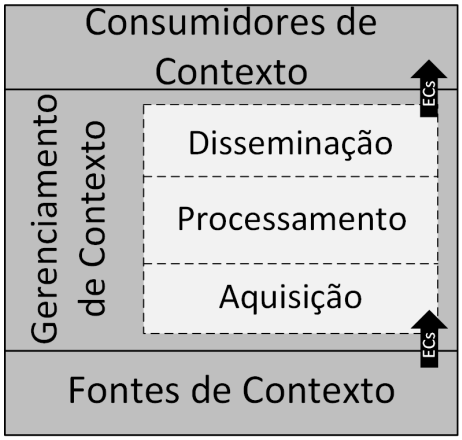


Figura 2.10: Modelo conceitual de arquitetura em camadas para ASCs.

A camada de Gerenciamento de Contexto exerce o papel da infraestrutura intermediária, sendo posicionada entre as fontes e os consumidores de contexto, de modo a fornecer os ECs adquiridos a partir destas fontes aos consumidores interessados. Nessa camada, o Módulo de Aquisição gerencia as fontes de contexto e recupera os ECs por meio do uso de componentes de software apropriados que permitem acesso às funcionalidades internas de cada fonte de contexto. O Módulo de Processamento é responsável por raciocinar e interpretar os ECs, transformando-os em um formato ou nível de abstração adequados para que sejam utilizados pelos consumidores de contexto. Além disso, esse módulo também pode realizar a combinação ou agregação de diferentes ECs provenientes de múltiplas fontes de contexto (BALDAUF; DUSTDAR; ROSENBERG, 2007). O Módulo de Disseminação, por sua vez, organiza os ECs e os disponibiliza aos consumidores de contexto. As Fontes de Contexto são elementos de software (e.g. perfis armazenados, bases de dados externas, drivers de câmeras e sensores) que fornecem informações atualizadas sobre as entidades consideradas no domínio da ASC. Por fim, os Consumidores de Contexto são elementos de software que usam os ECs relevantes para desempenhar algum comportamento sensível ao contexto. Tanto as fontes quanto os consumidores de contexto estão ligados à camada de gerenciamento de contexto através de interfaces

bem definidas (Dos Santos, 2008).

Perera et al. (2013) em seu estudo mostra que a maioria das arquiteturas em camadas de ASC existentes diferem entre si em aspectos relacionados às funcionalidades fornecidas, à localização e nomeação das camadas. Apesar disso, é possível identificar aspectos comuns na arquitetura das ASCs modernas quando se analisa as diferentes abordagens de projeto. Independentemente da organização dos módulos de uma ASC baseada em camadas, nota-se que a camada de Gerenciamento de Contexto é uma entidade que envolve a definição de soluções que permitem separar as operações relacionadas à manipulação de contexto das funcionalidades de negócio do domínio de aplicação (Dos Santos, 2008).

Capítulo 3

DESCOBERTA DE SERVIÇOS SENSÍVEL AO CONTEXTO

Ambientes de Computação Ubíqua geralmente envolvem diversos fatores que podem influenciar no desenvolvimento e execução de um sistema. Esses fatores podem se relacionar não apenas com o perfil do dispositivo de acesso, mas envolvem também as preferências e condições do usuário, as características da rede de acesso e do ambiente circundante, dentre outros fatores relevantes. Desta forma, para operar apropriadamente, uma aplicação ubíqua certamente deverá considerar em sua adaptação as variações que ocorrem tanto no perfil do dispositivo, como no perfil do usuário, da rede, do ambiente e assim por diante.

Nesse capítulo é apresentada a extensão do Ubiquitous Context Framework (UbiCon), um framework desenvolvido para apoio à construção de interfaces ricas adaptativas que se adaptam de forma híbrida, considerando os perfis de diferentes entidades contextuais envolvidas na execução da aplicação. Ele foi estendido para obter o contexto do perfil da rede de acesso, agregando mais uma fonte de contexto no framework, e possibilitando um maior número de informações do contexto em que se encontra. Também é apresentada a arquitetura do sistema que possibilita a descoberta de serviços sensível ao contexto.

3.1 Arquitetura Proposta

Uma abstração da arquitetura proposta é ilustrada na Figura 3.1. Esta arquitetura propõe a reutilização e a extensão do modelo arquitetural padrão de Web Services, acrescentando conhecimento e sensibilidade de contexto, obtido através de ontologias e do framework UbiCon Estendido (Seção 3.2), respectivamente. Ela possibilita a descoberta dinâmica de serviços sensível ao contexto. Apoiados por esta arquitetura, os serviços são publicados no UDDI Semân-

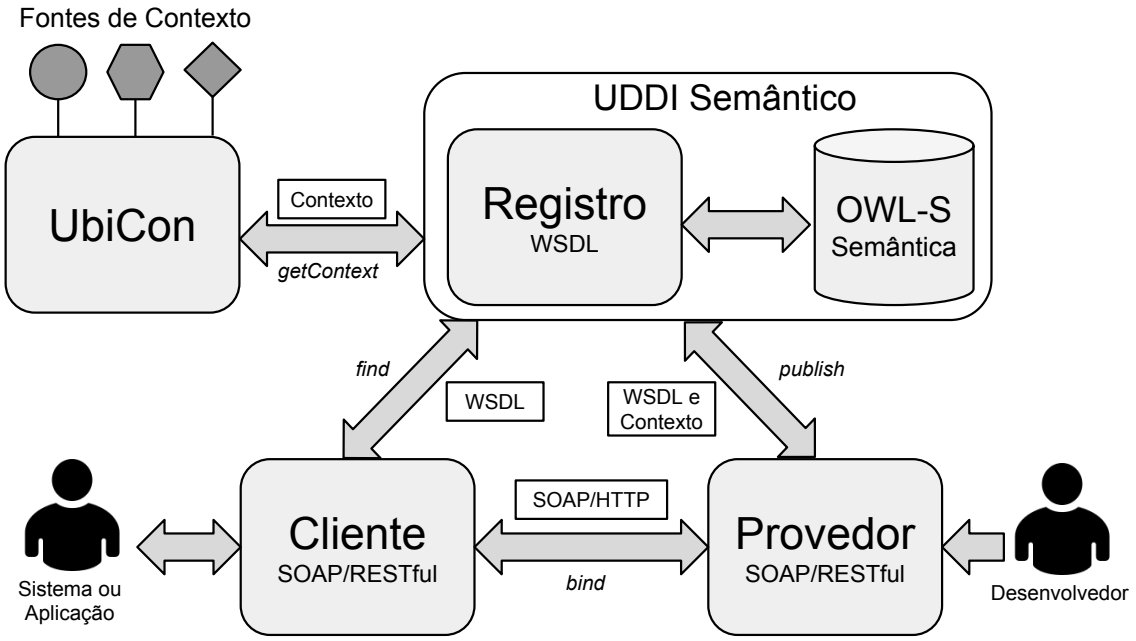


Figura 3.1: Modelo Arquitetural para Descoberta de Serviços Sensível ao Contexto

tico e disponibilizados concretamento no Servidor denominado aqui Provedor. No momento da publicação ocorre a conversão do serviço para ontologia OWL-S, que fica armazenada no UDDI Semântico, junto com a descrição WSDL do serviço. Assim, os serviços podem ser consumidos pelo Cliente. A descoberta dos serviços é apoiada pelo framework Ubicon Estendido, este responsável em descobrir o contexto atual da rede de acesso do Cliente. Além disso, os Web Services são buscados pela representação de seu conteúdo, através das ontologias construídas em tempo de execução, a partir da interpretação e validação de descritores de serviço. Desta maneira, estende-se o modelo padrão de WSs, para obter o contexto (UbiCon Estendido) e compará-lo com o conteúdo dos serviços (Ontologias), podendo ser interpretados dinamicamente pelo computador, alcançando o objetivo proposto. A seguir descreve-se com mais detalhes os principais componentes dessa arquitetura que apoio a descoberta dinâmica de serviços sensível ao contexto.

3.2 Framework UbiCon Estendido

Para desenvolver um sistema sensível ao contexto, muitos são os desafios. Pesquisas foram desenvolvidas com o propósito de gerar ferramentas e métodos específicos para apoiar o tratamento destes desafios, como, por exemplo, toolkits (DEY, 2000) e frameworks (Dos Santos, 2008; CIRILO et al., 2010). Essas abordagens objetivam suprir funcionalidades básicas relativas ao gerenciamento da informação contextual de forma que aplicações possam fazer uso dos serviços, simplificando o desenvolvimento destes sistemas. Para apoiar o gerenciamento do

contexto, foi desenvolvido, no mesmo grupo de pesquisa, um framework, nomeado Ubiquitous Context Framework (UbiCon) (CIRILO et al., 2010). Esse framework apoia o desenvolvimento de aplicações sensíveis ao contexto, abstraindo as funcionalidades relacionadas à manipulação do contexto e fornecendo serviços que manipulam o contexto.

A descoberta de serviços é uma tarefa dinâmica e varia de acordo com diferentes fatores. Portanto, a obtenção das informações contextuais necessárias para realizar a descoberta deve ser uma tarefa automática. Dessa forma, para recuperar as informações contextuais que orientam a descoberta de serviços, o framework UbiCon foi estendido, obtendo maior volume de informações contextuais a respeito das entidades (e.g. perfil da rede) associadas com as operações da descoberta de serviços. Foi escolhido o Perfil da rede, por ser o que tem maior influência na utilização dos serviços web. Dessa maneira, o mecanismo de descoberta pode consumir as informações contextuais obtidas pelo UbiCon Estendido, e realizar a descoberta dos serviços que satisfaçam as variações contextuais observadas.

3.2.1 Arquitetura do framework UbiCon Estendido

O framework UbiCon Estendido fornece aos desenvolvedores um "esqueleto"que pode ser instanciado para o desenvolvimento de aplicações ubíquas que adaptam-se de acordo com diferentes contextos. Ele encapsula as tarefas de manipulação de contexto, de forma que os desenvolvedores de software possam concentrar-se em aspectos mais importantes dos requisitos de negócio de suas aplicações. O UbiCon Estendido fornece funcionalidades divididas em 3 módulos: aquisição, processamento, disseminação, baseando-se na Arquitetura de Contexto proposta por Dos Santos (2008), citada anteriormente na subseção 2.3.3. A Figura 3.2 ilustra esses módulos e seus respectivos componentes.

Na Figura 3.2 são destacados os componentes implementados responsáveis pela obtenção do contexto. Os componentes que foram estendidos estão destacados em "sombreado", os que já existiam estão na cor branca. O UbiCon foi construído de forma extensível o suficiente para permitir que fontes de contexto que fornecem Elementos Contextuais (ECs) relacionados a tais perfis possam ser mais facilmente adicionados ao framework. Desta maneira, é possível estender o UbiCon para prover serviços baseados na combinação de diferentes perfis. As seções seguintes detalham a extensão do framework, módulos estendidos e os módulos da camada de gerenciamento do contexto.

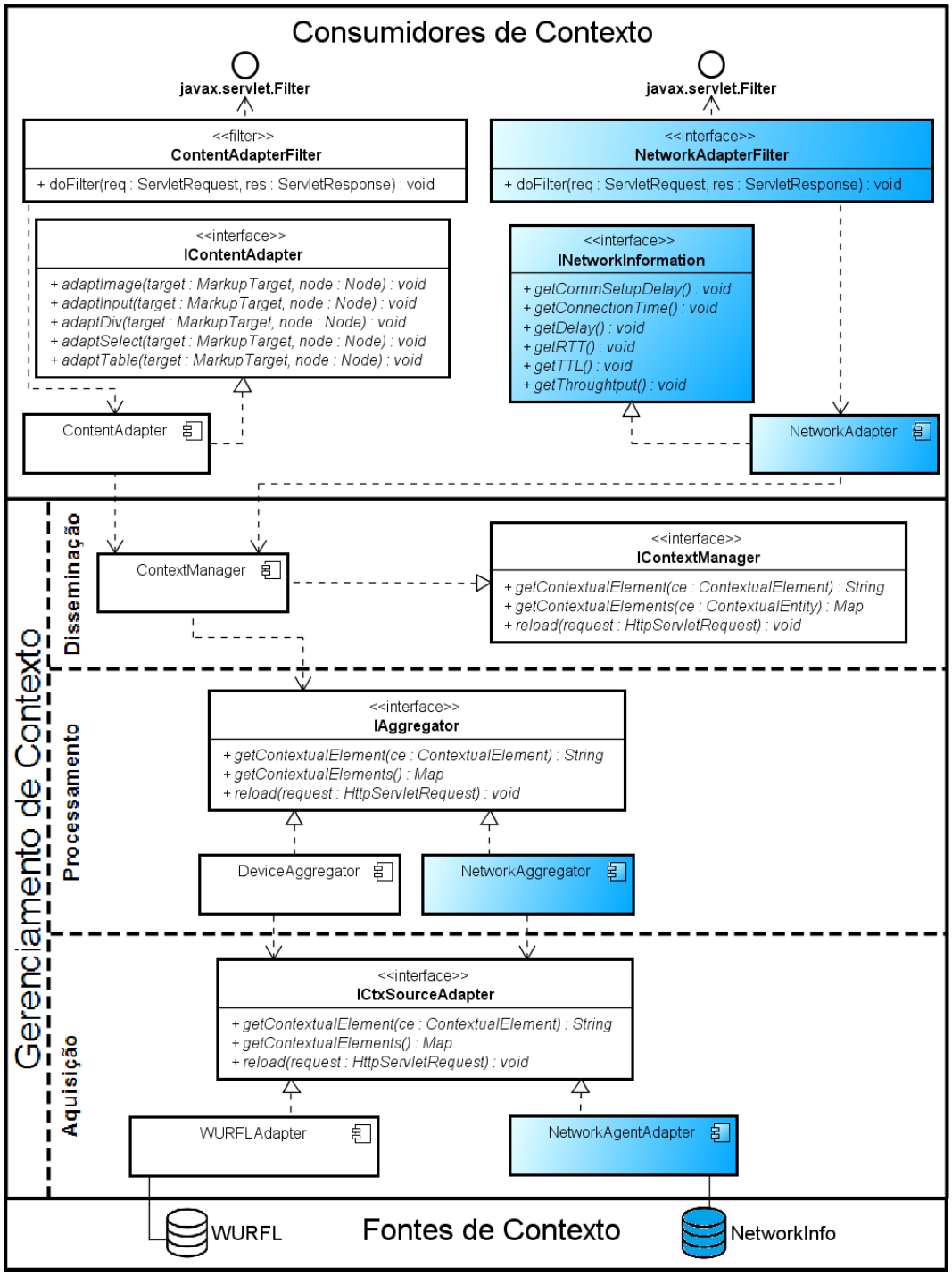


Figura 3.2: Modelo de Componentes do UbiCon Estendido

3.2.2 Extensão do UbiCon

O UbiCon foi projetado de forma a contemplar a obtenção do contexto de diferentes entidades envolvidas na execução de uma aplicação ubíqua. Essas são entidades do contexto computacional, por exemplo, características do dispositivo de acesso. Um dos perfis mais relevantes na descoberta de serviços é o perfil da rede de acesso do usuário. Dessa maneira, optou-se

por desenvolver primeiramente esse perfil. O perfil da rede é obtido dinamicamente pela classe *NetworkInfo* do UbiCon, a qual analisa a rede de acesso e a função do framework é monitorar os parâmetros da comunicação entre o cliente e o servidor. A informação contida nesse perfil (Figure 3.3) guia a descoberta de serviços.

A especificação das ontologias que caracterizam os perfis tem a base em trabalhos anteriores: *Extend Internet Content Adaptation Framework* (EICAF) (FORTE; SOUZA; PRADO, 2008). O EICAF é um framework de adaptação de conteúdo para aplicações Web baseadas em serviços web e ontologias. No EICAF, a informação contextual sobre as entidades de domínio (por exemplo, o dispositivo, usuário, SLA, rede, conteúdo) estão disponíveis através de perfis especificados em OWL. Essas ontologias foram refinadas e estendidas neste trabalho para atender aos requisitos de descoberta dinâmica de serviços e do contexto da aplicação.

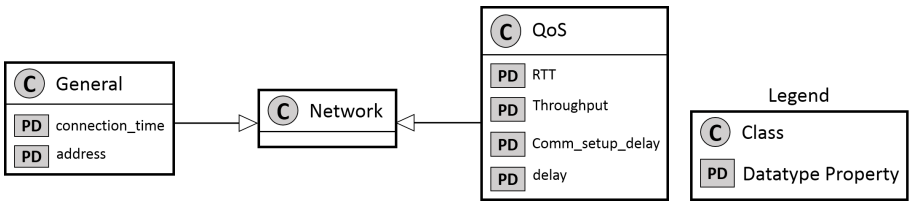


Figura 3.3: Modelo de ontologia que representa o perfil da rede (FORTE; SOUZA; PRADO, 2008)

A ontologia ilustrada na Figura 3.3, que representa o contexto computacional. Ela é criada em tempo de execução, com a ajuda da API Jena[1]. Os valores contextuais, da rede nesse caso, são capturados dinamicamente sempre que o contexto da rede sofre qualquer alteração, de modo que o contexto pode ser comparado com o contexto do serviço a ser descoberto, viabilizando a descoberta sensível ao contexto.

3.2.3 Módulos do Gerenciamento de Contexto

Dando ênfase para os componentes estendidos, os módulos da camada de gerenciamento de contexto englobam a aquisição dos ECs das fontes de contexto, o processamento e disseminação dos ECs. O módulo de aquisição compreende os componentes que acessam diretamente as fonte de contexto (e.g. bases de dados, drivers de sensores, agentes de software) para adquirir os ECs brutos, i.e., no formato entregue por estas fontes. Esses componentes são chamados de adaptadores (VIEIRA; TEDESCO; SALGADO, 2011), pois encapsulam os detalhes de acesso a fontes de contexto distintas, conectando o framework a fontes de contexto heterogêneas e fornecendo métodos genéricos para adquirir os ECs disponíveis. Esses métodos são disponibilizados através da interface **ICtxSourceAdapter** ilustrada na Figura 3.2.

[1] http://jena.apache.org/

A figura 3.4 mostra o diagrama de classes do Módulo de Aquisição implementado no UbiCon Estendido. No UbiCon a base de dados WURFL foi adotada como principal fonte de contexto para obtenção dos perfis do dispositivo. Já para o perfil da rede desenvolveu-se um agente de software que captura os dados da rede de acesso. Conforme mostrado na Figura 3.4, o componente chamado NetworkAgentAdapter acessa o agente desenvolvido, nomeado NetworkInfo, para obter os ECs associados ao perfil da rede de acesso.

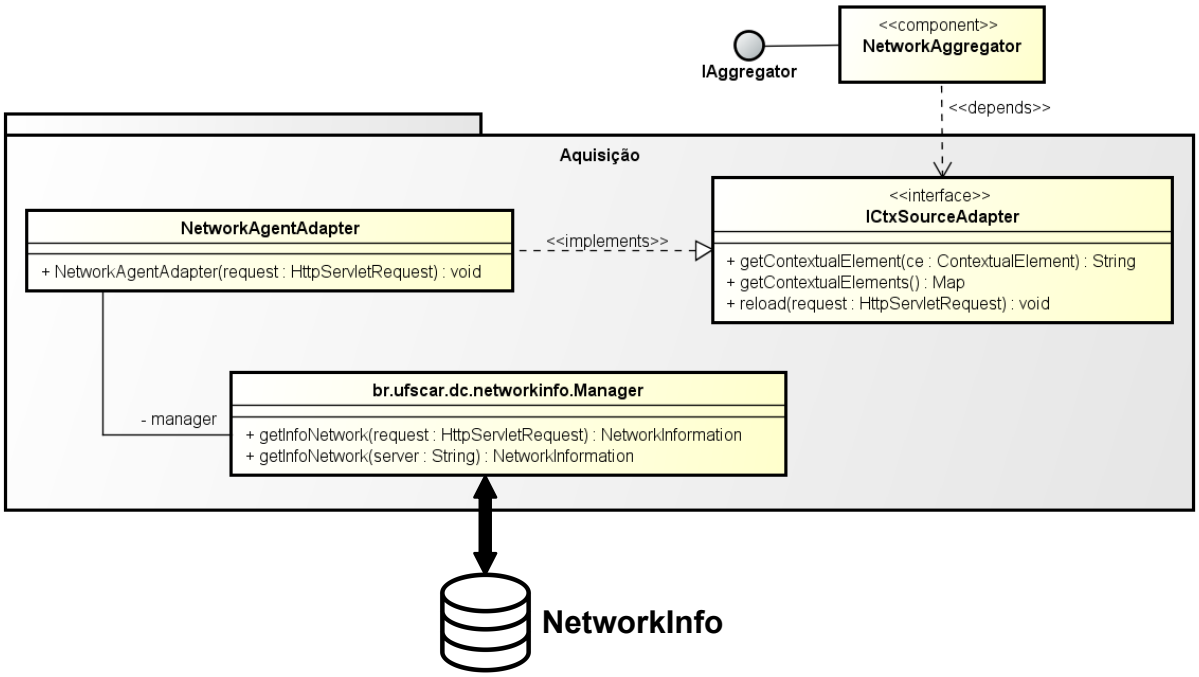


Figura 3.4: Modelo de Componentes do Módulo de Aquisição

Este módulo compreende os componentes que acessam diretamente as fontes de contexto para adquirir os ECs brutos, i.e., no formato entregue por estas fontes. Esses componentes são chamados de adaptadores (VIEIRA; TEDESCO; SALGADO, 2011), pois encapsulam os detalhes de acesso a fontes de contexto distintas, conectando o framework a fontes de contexto heterogêneas e fornecendo métodos genéricos para adquirir os ECs disponíveis. Esses métodos são disponibilizados através da interface ICtxSrcAdapter.

Já o módulo de processamento mostrado na figura 3.5 inclui os componentes agregadores que processam e agrupam os ECs de acordo com a entidade contextual que caracterizam. O processamento envolve a transformação dos ECs brutos em um formato mais adequado para ser utilizado pelos consumidores de contexto (BALDAUF; DUSTDAR; ROSENBERG, 2007). Cada agregador é então uma abstração de uma entidade contextual, fornecendo ECs refinados relacionados com a entidade a que está associado (DEY, 2001).

No framework UbiCon Estendido, os agregadores são construídos utilizando o padrão de

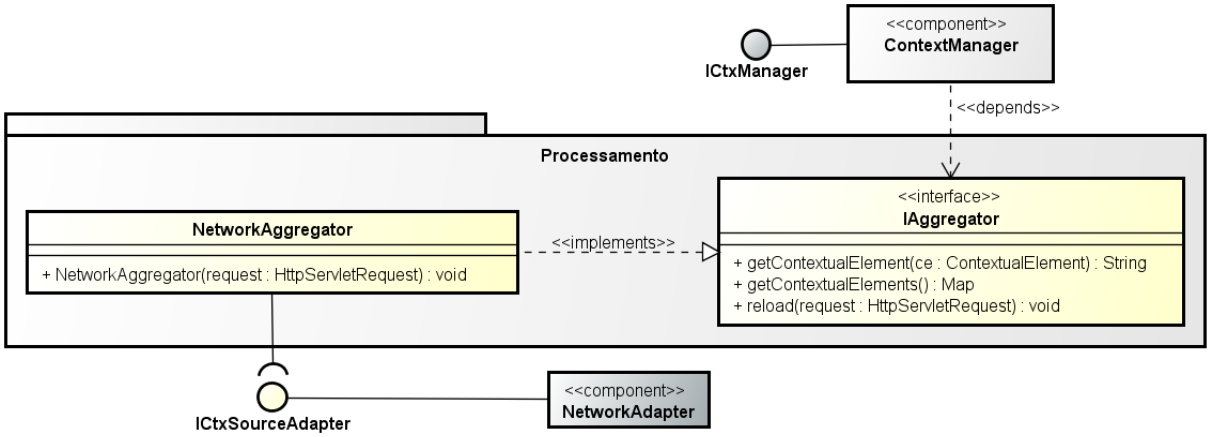


Figura 3.5: Modelo de Componentes do Módulo de Processamento

projeto denominado *Facade*. Assim, cada agregador atua como uma fachada que oculta a complexidade inerente à manipulação dos vários componentes do Módulo de Aquisição (adaptadores), fornecendo uma interface única para obtenção dos ECs de uma determinada entidade contextual.

O módulo de disseminação é composto por um componente principal, chamado ContextManager, o qual provê os ECs manipulados no Módulo de Processamento aos consumidores de contexto interessados através da interface ICtxManager, como mostra a Figura 2.3.2.

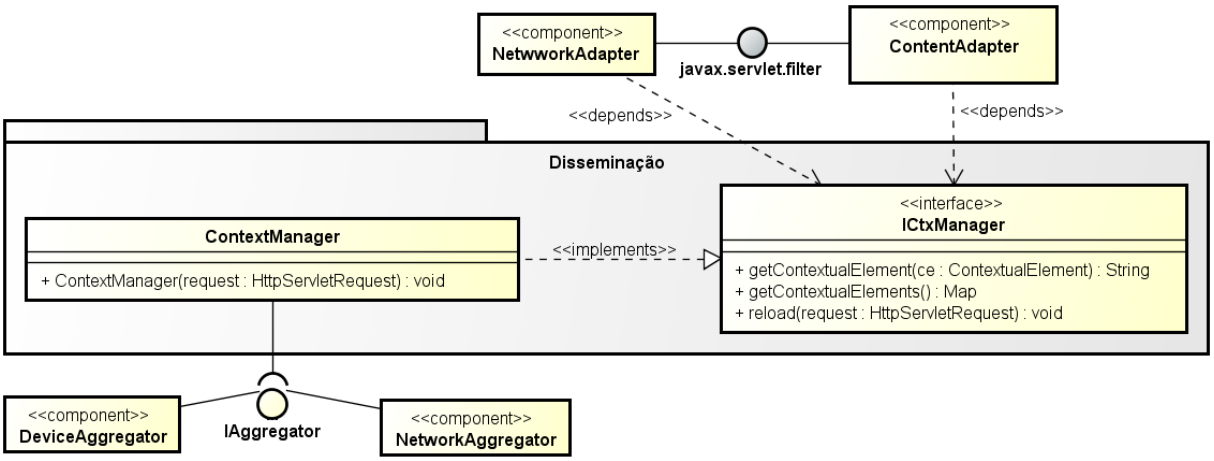


Figura 3.6: Modelo de Componentes do Módulo de Disseminação

Os consumidores de contexto podem requisitar o Módulo de Disseminação para obterem o valor atual de um EC específico ou todos os ECs relacionados a uma dada entidade contextual. A partir da perspectiva dos consumidores de contexto, o componente ContextManager atua como uma fachada, uma vez que oculta a manipulação dos diversos componentes do Módulo de Processamento (agregadores) e fornece uma única interface para obtenção dos ECs. A tarefa de determinar a partir de qual agregador requisitar um EC específico é encapsulada pelo ContextManager.

Devido a natureza extensível pela qual o UbiCon foi projetado, outras fontes de contexto podem ser adicionadas futuramente ao framework para complementar a aquisição dos ECs associados ao perfil do dispositivo. Isso implica na implementação de adaptadores específicos para acesso a estas fontes de contexto conforme ilustrado na Figura 3.4. Porém essa característica extensível simplifica a incorporação de fontes de contexto que fornecem ECs relacionados a outras entidades. Essas extensões permitem tornar o framework mais abrangente e possibilita fornecer diversos serviços sensíveis ao contexto.

Por fim, foi adicionado ao UbiCon uma nova fonte de contexto, nomeada NetworkInfo e novos módulos de aquisição, processamento e disseminação, assim como um novo consumidor de contexto, fornecendo informações contextuais da rede dinamicamente.

3.3 UDDI Semântico e Provedor

O UDDI Semântico é composto de um repositório de serviços e um repositório de ontologias. Ele foi desenvolvido a partir das especificações encontradas no site OASIS [2], e conta com o auxílio de um banco de dados relacional MySQL[3] para armazenar as informações de Provedores, Clientes e Serviços e um sistema de arquivos para armazenar o WSDL e seus respectivos OWL-S. A construção da arquitetura foi baseada na tecnologia Java EE[4] e Web Services baseados em RESTful para publicação e descoberta de serviços.

Para fazer a publicação de um serviço no UDDI Semântico, o **Provedor** necessita de um WSDL do serviço. No momento da publicação, demonstrado através do diagrama de sequência na Figura 3.7, a qual pode ser realizada de uma interface web do próprio UDDI ou por meio de requisição do Web Service de publicação.

O **Provedor** descreve algumas informações semânticas, por exemplo, o contexto em que o serviço melhor funciona, fornecendo como parâmetro as informações que formarão a ontologia mencionada anteriormente na Figura 3.3. Posteriormente, o UDDI fará a conversão de WSDL para OWL-S, e também armazenará as informações do serviço e seu contexto em um arquivo OWL, armazenando o WSDL original e mais duas ontologias, uma OWL-S com as informações IOPE do serviço e uma OWL com informações de contexto.

O UDDI Semântico atualmente suporta somente upload de descritores de serviços WSDL nas versões 2.0 e 1.1, realizando uma conversão para ontologia OWL-S em tempo de execução.

[2] https://www.oasis-open.org
[3] http://www.mysql.com/
[4] http://www.oracle.com/technetwork/java/javaee/overview/index.html

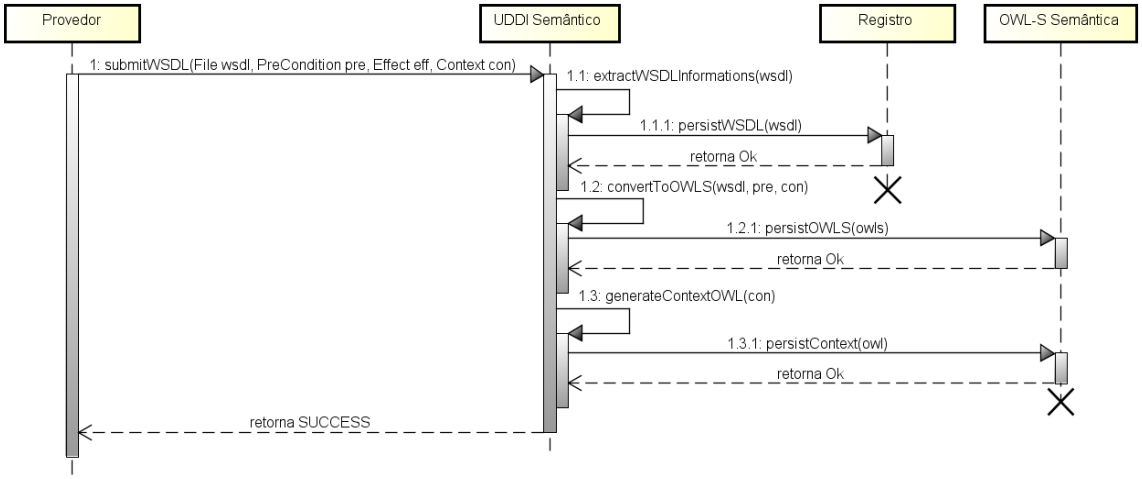


Figura 3.7: Diagrama de Sequência para publicação de serviços

Esta idéia foi proposa por Martin et al. (2004) e a conversão ocorre com o auxilio de APIs, como por exemplo, Jena[5], OWL-S[6] e Woden[7]. As APIs foram utilizadas para extração das informações relevantes dos arquivos WSDL. Essas informações são verificadas e filtradas, para gerar o arquivo OWL-S. Para cada serviço no arquivo WSDL, é gerado um arquivo OWL-S, o qual descreve semanticamente o serviço, que fica contido na classe *Service Profile* da ontologia OWL-S, como ilustra a visão abstrata do mapeamento realizado entre WSDL e OWL-S na Figura 3.8.

Este mapeamento de fato produz uma ontologia OWL, com os atributos de entrada, saída, efeitos e pré-condições (IOPE) de cada serviço publicado através do WSDL. Para auxiliar na descoberta dos serviços, a classe Service Profile da OWL-S pode conter regras especificadas em uma linguagem de regras para web semântica, neste caso a *Semantic Web Rule Language* (*SWRL*). As Figuras 3.9, 3.10, 3.11 e 3.12 ilustram as principais partes da especificação OWL-S da classe *Service Profile*, gerada a partir de um serviço.

Como um exemplo, neste caso, foi publicado um serviço que tem como objetivo fazer o *login* de um paciente na aplicação. Inicialmente são determinadas as ontologias a serem importadas, a fim de incluir a informação semântica já definida e necessária para este serviço, bem como, descrever uma relação entre uma instância de serviço e uma instância de perfil. Nesta ontologia, o serviço é identificado por *WSLogIn*, o qual é descrito pelo processo *WSLogInProcess* e representado pelo perfil *WSLogInProfile* na Figura 3.9.

Este código é seguido pela instância do *Service Profile* com os efeitos/resultados (1), pré-condições (2), saídas (3) e entradas (4) (Figura 3.10).

[5]https://jena.apache.org/
[6]http://on.cs.unibas.ch/owls-api/
[7]http://ws.apache.org/woden/

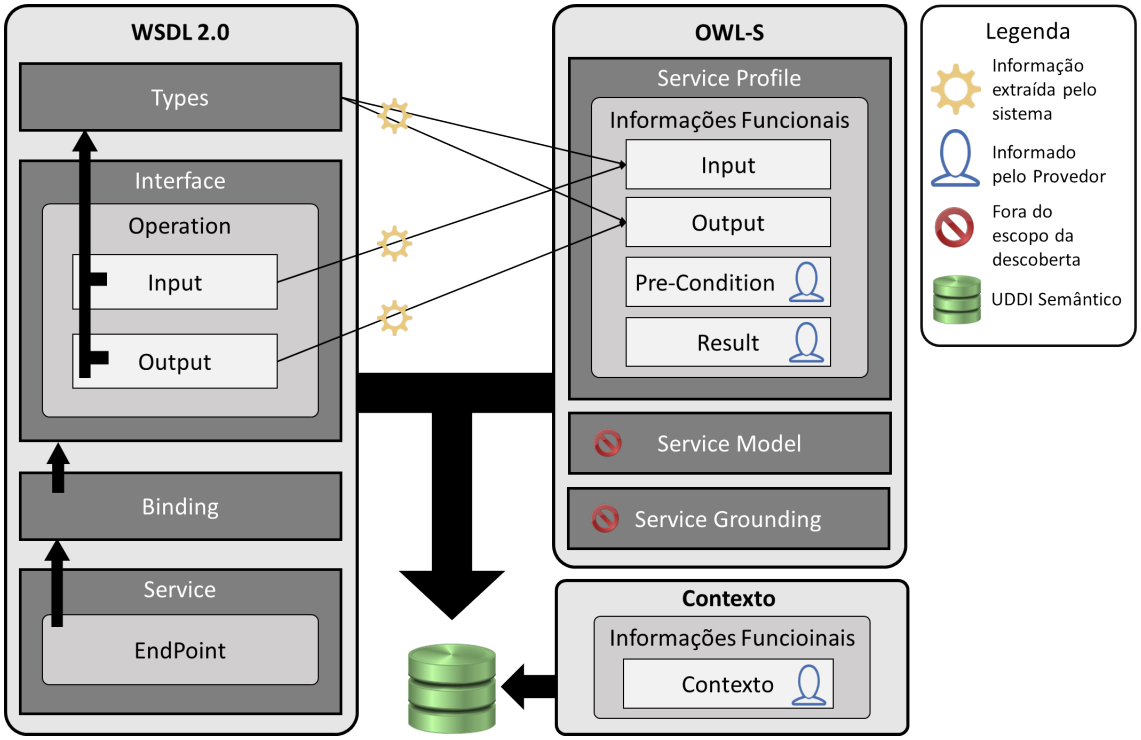


Figura 3.8: Visão Abstrata do Mapeamento(Baseada no trabalho de Farrag, Saleh e Ali (2013))

```
<service:Service rdf:about="WSLogIn">
        <service:describedBy>
                <process:AtomicProcess rdf:ID="WSLogInProcess"/>
        </service:describedBy>
        <service:presents>
                <profile:Profile rdf:ID="WSLogInProfile"/>
        </service:presents>
</service:Service>
```

Figura 3.9: Código fonte da ontologia gerada com informações que identificam o processo e o *Service Profile*

Nesta parte da ontologia, os atributos que são ilustrados permitem a descoberta semântica. Um componente essencial da classe *Service Profile* é a especificação de quais funcionalidades o serviço fornece, assim como a especificação das condições que devem ser satisfeitas para uma boa execução do mesmo. Este serviço tem como resultado o paciente logado na aplicação, seguro pela expressão SWRL com identificador *LoggedPatient*(1). A pré-condição *JoinedPatient* indica uma restrição de execução, a qual é verificada pela expressão SWRL que garante que o serviço é chamado somente se o paciente estiver registrado no banco de dados (2).

Na Figura 3.11 é mostrada a relação da instância do processo com as instâncias da descrição do serviço, que são representadas pela *Service Profile*. É importante compreender que um processo não é um programa a ser executado, mas é uma descrição de um serviço que espera uma mensagem e devolve outra mensagem de resposta. Um processo pode ter dois tipos de propó-

```
<profile:Profile rdf:about="#WSLogInProfile">
  <service:presentedBy rdf:resource="WSLogIn"/>
  <profile:hasResult>
        <process:Result rdf:ID="LoggedPatient">
              <process:hasEffect rdf:resource="#LoggedPatient"/>
              <expr:expressionLanguage rdf:resource="#SWRL"/>
              <rdf:type rdf:resource="#SWRL-Expression"/>
        </process:Result>
  </profile:hasResult>
  <profile:hasPrecondition>
        <expr:SWRL-Condition rdf:ID="JoinedPatient">
              <expr:expressionObject rdf:resource="#nil"/>
              <expr:expressionLanguage rdf:resource="#SWRL"/>
        </expr:SWRL-Condition>
  </profile:hasPrecondition>
  <profile:hasOutput>
        <process:Output rdf:about="result">
              <process:parameterType rdf:datatype="#anyURI">boolean</process:parameterType>
        </process:Output>
  </profile:hasOutput>
  <profile:hasOutput>
    <process:Output rdf:about="patient">
        <process:parameterType rdf:datatype="#anyURI">object</process:parameterType>
        <rdf:type rdf:resource="#Input"/>
    </process:Output>
  </profile:hasOutput>
  <profile:hasInput rdf:resource="patient"/>
</profile:Profile>
```


Figura 3.10: Código fonte da ontologia gerada com informações sobre IOPE

sito. Primeiro, ele pode gerar e retornar alguma informação nova com base na informação que é dada, e segundo, ele pode produzir uma mudança. Esta mudança é descrita pelas condições e os efeitos do processo. Um processo pode ter qualquer número de entradas (incluindo zero), que representa a informação que é, em algumas circunstâncias, apropriada para a realização do processo. Ele pode ter qualquer número de saídas, a informação de que o processo fornece ao solicitante. Pode haver qualquer número de condições, as quais devem garantir que o processo seja invocado com sucesso. Finalmente, o processo pode ter qualquer número de efeitos. Saídas e efeitos pode depender de condições que possuem verdadeiro no estado no momento em que o processo é realizado.

A pré-condição *JoinedPatient* é baseada na regra SWRL da Figura 3.12, a qual verifica se o paciente está registrado no sistema, antes da execução do serviço, retornando uma resposta *true* quando o paciente se encontra no banco de dados, permitindo a continuação do fluxo de descoberta. Se um processo tem uma pré-condição, então o processo não pode ser realizado com sucesso a menos que a condição seja verdadeira.

```
<process:AtomicProcess rdf:about="#WSLogInProcess">
        <service:describes rdf:resource="WSLogIn"/>
        <process:hasResult rdf:resource="#LoggedPatient"/>
        <process:hasPrecondition rdf:resource="#JoinedPatient"/>
        <process:hasOutput rdf:resource="result"/>
        <process:hasOutput rdf:resource="patient"/>
        <process:hasInput rdf:resource="patient"/>
</process:AtomicProcess>
```

Figura 3.11: Código fonte da ontologia gerada com informações referentes a identificação do processo e suas devidas instâncias

```
<swrl:AtomList>
    <rdf:first>
        <swrl:IndividualPropertyAtom>
            <swrl:argument2>
                <swrl:Variable rdf:ID="joined" />
            </swrl:argument2>
            <swrl:argument1>
                <swrl:Variable rdf:ID="patient" />
            </swrl:argument1>
            <swrl:propertyPredicate>
                <owl:ObjectProperty rdf:about="PatientState" />
            </swrl:propertyPredicate>
        </swrl:IndividualPropertyAtom>
    </rdf:first>
</swrl:AtomList>
```

Figura 3.12: O código-fonte da ontologia gerada com informações sobre a regra da pré-condição

3.4 Cliente

A Descoberta de Serviços sensível ao contexto é o principal objetivo desta pesquisa. Ela acontece quando o **Cliente** que previamente possui uma composição de serviços, possui algum serviço da sua composição que não atende sua necessidade, situação ou seu contexto. Para cada serviço do repositório existe um arquivo descrevendo o contexto do serviço, este representado pela ontologia ilustrada na Figura 3.3 da Seção 3.2. Sempre que o UbiCon detecta a mudança no contexto do serviço, o sistema verifica se a composição inicial satisfaz o contexto atual do serviço, se não satisfazê-lo, ele inicia a busca por um outro serviço no UDDI Semântico. O serviço de busca é sempre baseado em uma composição inicial de serviços.

Suponha que haja uma composição de três serviços S1, S2 e S3. Nesta composição o serviço S2 não atende o contexto atual do cliente. Após o sistema de descoberta perceber a mudança no contexto, ele inicia uma busca utilizando como parâmetro de entrada o descritor

do serviço que se encontra fora do contexto. A execução do algoritmo se inicia, convertendo o descritor de serviço (WSDL) em OWL-S, com auxílio da API da OWL-S [8], obtendo desta maneira os dados de entrada, saída, pré-condições e efeitos da classe *ProfileService*, que estende a classe *Service* da ontologia OWL-S. A partir destas representações os serviços são comparados com todos os outros serviços do repositório por meio de inferências. Uma lista com os possíveis serviços que substituirão o S2 é retornada ou uma lista vazia no caso de insucesso do algoritmo ou ainda se nenhum serviço do repositório satisfaz o contexto do cliente.

O Código 3.1 mostra o algoritmo, escrito em pseudo-código, que é executado para descoberta de serviços sensível ao contexto.

Code 3.1: Algoritmo da Descoberta Dinâmica de Serviços Sensível ao Contexto

```
function List<Service> discovery(service_outside_context)
  List<Service> servicesDiscovery;
    context = ubicon.context;
      if (context.change)
        for each WS s in UDDI
          if (s.IOPE == service_outside_context.IOPE)
            for each attributes in context
              if (context.attribute <= s.context.attribute)
                servicesDiscovery.add(s);
              else
                servicesDiscovery.remove(s);
  return servicesDiscovery;
```

A descoberta fornece busca semântica através de conteúdos e não apenas por palavras-chave, facilitando a compreensão pelo computador (MARTIN et al., 2004). A descoberta de WSs depende do descritor de serviço, neste caso, o WSDL, o qual complementa o padrão UDDI, fornecendo uma maneira uniforme para descrever a interface e a conexão entre Provedor e o Cliente. Como as especificações UDDI não definem um processamento semântico para o WSs, o UDDI deve ser estendido, a fim de entender o contexto semântico do serviço, permitindo desta forma a descoberta dinâmica de serviços semânticos dentro do UDDI. A descoberta de serviços sensível ao contexto acontece quando o cliente deseja invocar algum tipo de serviço que atenda a sua situação ou contexto em que ele se encontra, como mostra a Figura 3.13.

Sempre que o sistema detecta a mudança no contexto, ele verifica se a composição inicial satisfaz o contexto atual, se não satisfazê-lo, começa a busca por um outro serviço no UDDI Semântico. O serviço de descoberta é sempre baseado em uma composição inicial de serviço. Suponha que haja uma composição de três serviços WS1, WS2 e WS3. Nesta composição

[8] http://on.cs.unibas.ch/owls-api/

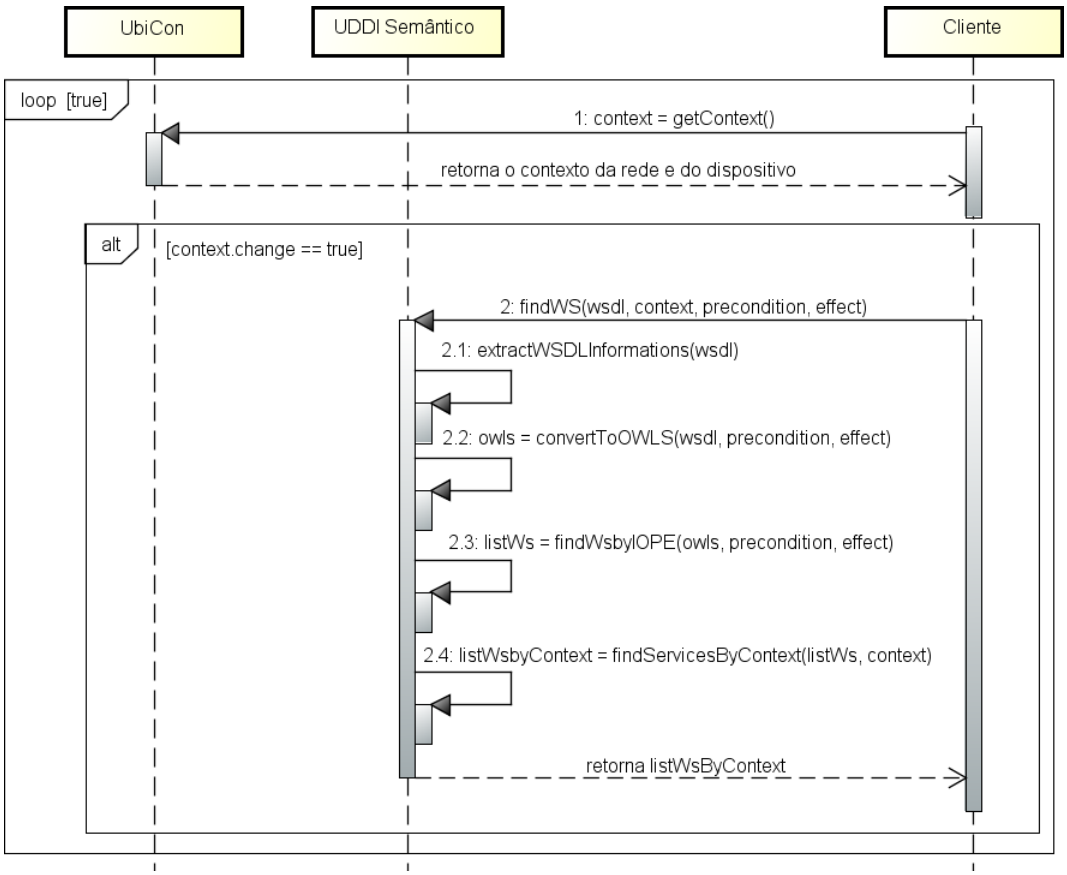


Figura 3.13: Diagrama de sequencia para publicação do WSDL pelo Provedor

o WS2 não atende mais o contexto atual do cliente. Usando os dados de entrada, de saída, pré-condições e efeitos da classe *Service Profile*, a qual estende a classe *Service* da ontologia OWL-S, o sistema inicia uma busca convertendo o descritor em ontologia. Através dessas representações, a informação e os serviços de contexto são comparados para todos os WSs disponíveis no UDDI Semântico, por meio de inferências. Assim, ele retorna uma lista de possíveis WSS para substituir o WS2. Se não existem serviços disponíveis, ele retorna uma lista vazia. Assim, tem-se a descoberta de serviços sensível ao contexto.

3.5 Considerações Finais

Esse capítulo apresentou a proposta e o que foi desenvolvido nesta pesquisa. Primeiramente foi desenvolvida uma arquitetura, que reutiliza a arquitetura padrão de web services, que já é bem conhecida e aceita pela comunidade acadêmica e pela indústria de software. A partir desta arquitetura, foi estendido o framework UbiCon, agregando a fonte de contexto de rede, está que por sua vez é essencial no processo de descoberta dos serviços de acordo com o contexto. Para execução mais eficiente desse processo de descoberta, foi utilizado o padrão WSDL2, que

descreve serviços RESTful e SOAP, agregando a descoberta esses dois tipos de serviços. Por fim foi desenvolvida uma avaliação que é apresentada no capítulo seguinte.

Capítulo 4

AVALIAÇÃO

Avaliação é um processo altamente crítico para a melhoria da qualidade de produção de software. Através da avaliação é que se pode verificar a efetividade de novas ferramentas, métodos, processos e estratégias de desenvolvimento de software antes de colocá-los em prática. Se nenhuma avaliação é realizada, a introdução de uma nova tecnologia no desenvolvimento de software pode ser ineficiente e até mesmo ter impactos negativos na qualidade do software desenvolvido. A avaliação na Engenharia de Software, portanto, fornece bases para validar as teorias a respeito de uma tecnologia e confrontá-las com a realidade, com o intuito de descobrir se irão, de fato, surtir o efeito esperado tanto no produto quanto no processo (TRAVASSOS; GUROV; AMARAL, 2002; WOHLIN et al., 2012).

O desenvolvimento de software é, na maioria das vezes, uma tarefa complexa. Projetos de software podem se estender por longos períodos de tempo, envolver diferentes pessoas com habilidades distintas, e consistir de várias atividades que produzem diversos documentos intermediários para especificação do software a ser entregue. Esta complexidade acarreta em dificuldades para a otimização dos processos de desenvolvimento (WOHLIN et al., 2012). Neste sentido, a avaliação e o aperfeiçoamento dos processos de software têm sido estratégias aplicadas ao longo dos anos em busca da melhoria da qualidade do software desenvolvido. Em geral, os métodos de avaliação baseiam-se na premissa de que a qualidade do produto de software é determinada pela qualidade de seu processo de desenvolvimento, considerando que um bom produto é resultado direto de um bom processo (TEAM, 2006; CASAL et al., 1998).

De fato, a literatura sugere que novas abordagens de desenvolvimento de software devem ser avaliadas no sentido de verificar sua efetividade antes de colocá-las em prática (TRAVASSOS; GUROV; AMARAL, 2002; WOHLIN et al., 2012). Desta maneira, neste capítulo, é apresentada a avaliação da Descoberta de Serviços Sensível ao Contexto, com base em um estudo de caso apresentado a seguir.

4.1 Avaliação de Desempenho

Como forma de avaliar a arquitetura proposta, bem como averiguar o comportamento das ontologias e a extensão do framework UbiCon, um estudo de caso foi conduzido desenvolvendo uma aplicação no domínio de Marcação de Consultas Médicas e Atendimento Médico.

Com objetivo de avaliar a capacidade de carga e desempenho do UDDI Semântico e do framework UbiCon Estendido, que são os principais componentes da arquitetura proposta, um estudo de caso foi realizado envolvendo dois computadores em um ambiente de testes, tal como é ilustrado na Figura 4.1. O computador (1) foi configurado com Windows 7 (2 Núcleos 2.27GHz/4GB) e o (2) foi configurado com Linux Ubuntu 13.04 Server (4 Núcleos 3.3GHz/12GB).

No computador (1) foi instalada e configurada uma aplicação *Desktop* chamada Apache JMeter[1], a qual simula um ou mais usuários acessando o sistema simultaneamente. Na parte servidora (2), foi instalado o UDDI Semântico que utiliza o UbiCon Estendido como fonte de contexto e configurado um servidor Apache Tomcat Server 7[2] com *hosts* virtuais para serem acessados. Isso permite os testes realizados serem copiados e comparados.

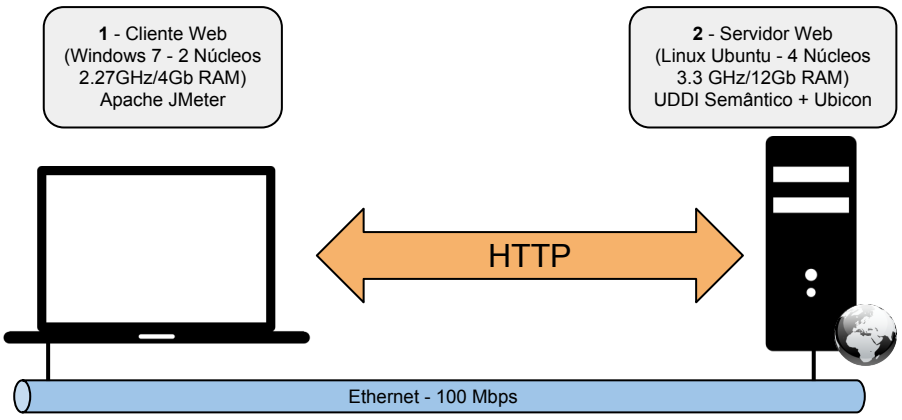


Figura 4.1: Ambiente de Testes para avaliar a instância arquitetura proposta

O teste consistiu em acessar repetidamente os serviços web do UDDI Semântico, tanto serviços de *upload* dos WSs, quanto serviços de busca e descoberta de WSs. Para acessar estes serviços, foi aplicada uma carga de acessos crescentes ao UDDI, chegando ao limite de 100 usuários simultâneos, com 100 contextos diferentes, gerados randomicamente pela aplicação. Os usuários, por sua vez, realizavam 25 requisições com intervalos de 10 segundos entre elas, totalizando 2500 requisições em cada teste. Esta foi a configuração feita no JMeter para o ambiente de testes. Desta forma, foram definidos 3 cenários de testes. No primeiro cenário,

[1]http://jmeter.apache.org
[2]http://tomcat.apache.org

foram realizados testes relacionados a inserção dos serviços no repositório, em outras palavras, *uploads* dos *WSDLs* no UDDI Semântico, este realizando as devidas validações e conversões entre descritores de serviços e ontologias relacionadas. No segundo cenário, foram realizados teste de busca de serviços, sem o contexto e sem ontologias, somente através dos WSDLs. E num terceiro cenário, foram realizadas buscas de serviços por contexto.

No teste de inserção, o arquivo WSDL é passado como parâmetro, assim como o contexto, em que o serviço funciona melhor. Através dos atributos definidos no arquivo, e o contexto passado como parâmetro, é invocado outro serviço que gera as ontologias OWL-S e a respectiva ontologia do contexto.

Para o teste de busca de serviços sem ontologia, ocorre uma busca por palavras-chave, que estão vinculadas ao serviço, através dos métodos e atributos de entrada e saída.

Já no teste que busca através de contextos e inferências das ontologias, foram passados como parâmetro contextos diversos, gerados aleatoriamente e randomicamente, e serviços que estavam em uma composição de serviços e já não satisfaziam o contexto atual. Esses serviços são convertidos para ontologias e são realizadas inferências sobre o contexto e o IOPE do serviço, como apresentado anteriormente no algoritmo 3.1, descrito na Seção 3.4.

Os resultados dos testes de capacidade de carga e tempo de respostas são apresentados na Figura 4.2 e 4.3. Na figura 4.2 é ilustrado um gráfico de área empilhada. No eixo X é mostrado o numero de execuções do teste, enquanto o eixo Y mostra o percentual de possíveis WSs a serem descobertos. O possíveis WSs são os que tem os mesmos atributos IOPEs do WS que está fora do contexto. A maior área refere-se a serviços retornados que estão dentro do contexto do usuário, e a menor área refere-se aos serviços que não estão dentro de um contexto apropriado, validando a proposta de sensibilidade ao contexto. Falando em porcentagens, tem-se uma média de 94,8% de acertos contra 5,2% dos serviços não encontrados dentro daquele contexto.

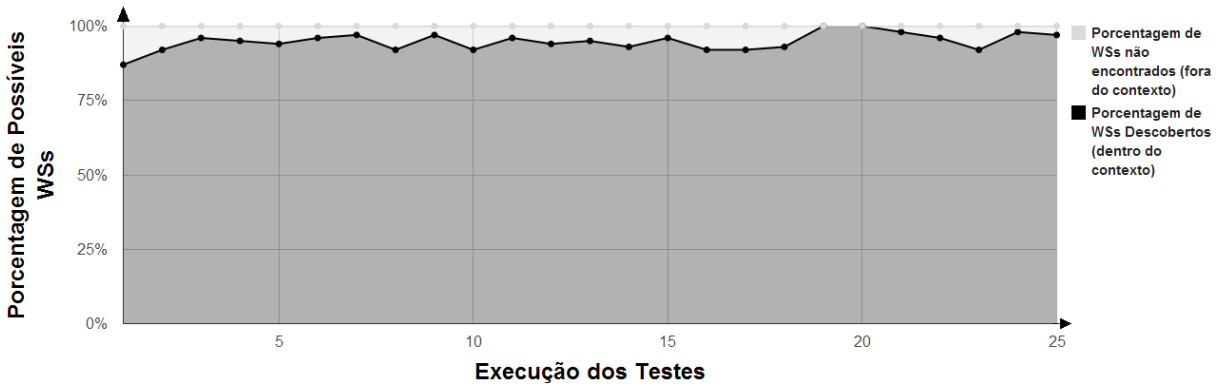


Figura 4.2: Sensibilidade ao Contexto: Porcentagem de possíveis serviços vs número de execuções de teste

A Figura 4.2 ajuda a perceber a pequena variação no contexto da rede que foi obtida neste ambiente e com estas configurações, para a execução do algoritmo de descoberta dinâmica de serviços. A variação do contexto da rede é ilustrada pela linha com pontos pretos, concluindo que a descoberta de serviços é sensível ao contexto.

Uma análise quantitativa dos resultados é ilustrada na Figura 4.3, como gráfico de linha, o qual indica um aumento no tempo de resposta e na demanda de processamento causada pela extração e conversão de informações do arquivo WSDL em ontologias, bem como a criação e uso de ontologias, convertidas em tempo de execução para o contexto, fornecido pelo UbiCon Estendido. No entanto, do ponto de vista qualitativo, deve-se levar em conta uma maior flexibilidade com um aumento de precisão, resultante da utilização semântica no mecanismo de descoberta, o qual facilita o trabalho de desenvolvedores de serviços web dando a ele um alto grau de satisfação na descoberta de serviços dinamicamente. Além disso, este estudo de caso indica que a opção de realizar a descoberta de serviços próprio resultou em um ganho significativo para a composição de serviços em relação a outros estudos de casos em que utilizaram UDDI adaptados.

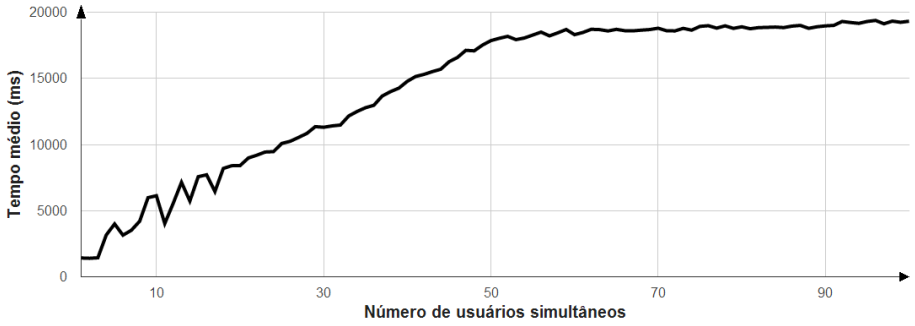


Figura 4.3: Tempo médio de resposta do servidor vs número de usuários executando o serviço de descoberta ao mesmo tempo

4.2 Experimentação

Na Engenharia de Software a experimentação é importante para testar hipóteses sobre uma nova tecnologia, observando se os efeitos da sua adoção estão alinhados com o que foi declarado nas hipóteses (WOHLIN et al., 2000). A experimentação, portanto, verifica a previsão teórica de encontro à realidade, de forma a prover evidências de que o objeto avaliado (arquitetura) está ou não indo na direção esperada (TRAVASSOS; GUROV; AMARAL, 2002).

A experimentação da arquitetura proposta foi conduzida no final do primeiro semestre de 2014. O experimento consistiu de um estudo comparativo entre a utilização dos componentes da arquitetura proposta e os componentes da arquitetura padrão de WSs, utilizando como registro

a ferramenta jUDDI. O experimento foi conduzido em relação a publicação e descoberta de serviços.

Para a realização do experimento foi utilizada a documentação de uma aplicação ubíqua com objetivo de facilitar e agilizar a marcação de consultas médicas pela Internet, elaborada especialmente para a execução do experimento. Essa aplicação, nomeada HSS (Health Service System), é constituída por serviços web, logo, esses serviços ficam armazenados no UDDI Semântico, que representa a parte Servidora da aplicação.

A tarefa dos participantes durante o experimento foi desenvolver versões diferentes dos serviços, baseando-se em documentos e diagramas de Casos de Uso e diagramas de Classes, formando uma base com diversos serviços semelhantes. Os participantes desenvolveram três serviços, os quais faziam uma pequena composição entre suas entradas e saídas. Após este desenvolvimento os serviços foram publicados, tanto no UDDI Semântico, quanto no jUDDI.

Durante a execução do experimento, os participantes registraram, em um formulário entregue a eles, a hora de início e fim da realização de cada atividade executada, bem como as quantidades de linhas de código geradas manualmente. Na contabilização das linhas de código, foi considerado apenas o código referente aos serviços escritos em WSDL. Cada participante recebeu um material de apoio apropriado com diretrizes que os orientou nas atividades de desenvolvimento dos serviços.

As fases experimentais realizadas no estudo são detalhadas nas subseções seguintes.

4.2.1 Definição do Experimento

O objetivo do estudo foi:

- Analisar a arquitetura proposta, instanciando seus componentes para publicação e descoberta de serviços, com o propósito de avaliação, com respeito à eficiência em termos de tempo despendido e produtividade, do ponto de vista de desenvolvedores de software, no contexto de estudantes de pós-graduação em Ciência da Computação.

O contexto de um experimento pode ser caracterizado em termos do número de participantes e objetos envolvidos no estudo experimental. No experimento realizado, considerando o envolvimento de diversos participantes e de um objeto (aplicação HSS), o contexto do experimento classificou-se como estudo de objeto com vários testes (WOHLIN et al., 2000). Isso significa que o estudo consistiu de diversos testes experimentais, onde em cada teste houve

um participante que desenvolveu um determinado conjunto de serviços, como proposto pelo experimentador.

4.2.2 Planejamento do Experimento

O experimento ocorreu em ambiente universitário, sendo realizado com estudantes de pós-graduação no Departamento de Computação da Universidade Federal de São Carlos (UFSCar). A seleção dos participantes foi feita através de amostragem não probabilística por conveniência (WOHLIN et al., 2000), selecionando os indivíduos disponíveis mais próximos para participarem do experimento. Participaram do estudo 10 alunos de Pós-Graduação em ciência da Computação da UFSCar, os quais possuem certa experiência com desenvolvimento de software e conhecimento acerca das tecnologias e ferramentas utilizadas no estudo. A experiência dos participantes foi avaliada pelo Formulário de Caracterização dos Participantes (Apêndice A) - documento utilizado para capturar a experiência profissional e experiência nos assuntos relacionados ao estudo (e.g. XML, OWL, WSDL, UDDI).

Foram elaboradas três hipóteses para o experimento com relação ao efeito da publicação e descoberta de serviços web. Para formulação das hipóteses foram consideradas as seguinte métricas:

1) t - Tempo total gasto pelo participante para planejamento, desenvolvimento e testes dos Web Services;

2) p - Produtividade do participante em termos de linhas de código produzidas (LOC) por unidade de tempo (p = LOC/t);

3) Mt - Tempo médio despendido pelos participantes para o desenvolvimento dos Web Services;

4) Mp - Produtividade média dos participantes para o desenvolvimento dos Web Services;

Hipótese nula (H_0): Em geral, não há diferença entre a arquitetura proposta e a arquitetura padrão de WSs, uma vez que os participantes utilizando as duas ferramentas não notaram grandes diferenças.

H_0: $E_{proposta} = E_{padrao}$ –> $Mt_{proposta} = Mt_{padrao}$ e $Mp_{proposta} = Mp_{padrao}$

Hipótese alternativa (H_1): Participantes utilizando a arquitetura proposta são mais eficientes do que participantes utilizando a arquitetura padrão de WSs com jUDDI.

H_1: $E_{proposta} > E_{padrao}$ –> $Mt_{proposta} < Mt_{padrao}$ e $Mp_{proposta} > Mp_{padrao}$

Hipótese alternativa (H_2): Participantes utilizando a arquitetura padrão de WSs com jUDDI são mais eficientes do que participantes utilizando a arquitetura proposta.

H_2: $E_{proposta} < E_{padrao} -> Mt_{proposta} > Mt_{padrao}$ e $Mp_{proposta} < Mp_{padrao}$

A variável dependente selecionada para o experimento foi a eficiência (E) dos participantes. As variáveis independentes foram a arquitetura usada para publicação e descoberta de WSs, a aplicação desenvolvida, o ambiente de desenvolvimento e as tecnologias de desenvolvimento. Uma vez que o objetivo era investigar as consequências da adoção de uma determinada arquitetura na eficiência dos participantes, a arquitetura de publicação e descoberta de WSs foi definida como como o fator que receberia dois tratamentos distintos (UDDI Semântico e jUDDI) e cujo efeito deveria ser observado na variável E.

Os materiais necessários para apoiar os participantes no experimento foram previamente planejados, compreendendo a definição dos objetos que seriam manipulados, as diretrizes para orientar os estudantes na execução das tarefas, bem como os instrumentos para a coleta de dados e medição.

4.2.3 Execução do Experimento

A execução do experimento foi realizada em três passos:

1) Preparação. Nesta fase os materiais definidos na instrumentação do experimento foram efetivamente elaborados, a saber:

- **Objetos.** O UDDI Semântico, foi implementado com Web Services de publicação e descoberta, assim como uma interface web, juntamente com parte do framework UbiCon Estendido para reconhecimento do contexto da rede. O projeto estava disponível em um servidor para que os participantes pudessem executar as tarefas de publicação e descoberta. O repositório jUDDI também foi disponibilizado assim como o UDDI Semântico.

- **Diretrizes.** Os seguinte documentos foram produzidos para serem utilizados no estudo: 1) *Formulário de caracterização dos participantes*, para obtenção da experiência profissional e nos assuntos relacionados diretamente ao estudo; 2) *Formulário de consentimento*, para aprovação e ciência dos participantes dos objetivos do estudo e das condições de participação; 3) *Descrição da tarefa*, com as instruções da sua execução; e 4) *Descrição da aplicação e material de apoio*, contendo o detalhamento e exemplificação da aplicação HSS, bem como um tutorial para publicação e descoberta dos Web Services;

- **Instrumentos para coleta de dados.** Medições em experimentos são conduzidas atra-

vés dos dados que são coletados. Em experimentos envolvendo atividades executadas por seres-humanos, os dados geralmente são coletados manualmente por meio de formulários ou entrevistas. Assim, para a coleta de dados do experimento, foi elaborado o *Formulário de Coleta de Dados*, no qual os participantes deveriam preencher todas as informações requeridas durante a execução do experimento, relatando tempos, problemas e LOC. Além disso, foi elaborado um *Formulário de Avaliação*, que avalia qualitativamente a arquitetura proposta, em termos de facilidade de utilização e contém questões relacionadas a diferenças entre as arquiteturas.

Todos os documentos elaborados para instrumentação do estudo encontram-se anexados a esta dissertação nos Apêndices.

Para evitar que o tempo de aprendizagem de como utilizar os componentes da arquitetura proposta interferisse no tempo de desenvolvimento do experimento, foram realizados treinamentos como forma de aquecimento (*warm-up*) com todos os participantes do experimento um pouco antes de realizar o experimento em si.

2) Execução. Os participantes executaram os testes experimentais conforme projeto experimental. Na tabela 4.1 são mostrados os dados coletados pelos participantes na execução do experimento no que tange ao projeto e implementação dos WSs.

Participante	Projeto			Implementação			Testes-UDDISemântico			Testes-jUDDI		
P1	17:17	17:21	**00:04**	17:21	17:39	**00:18**	17:39	17:43	**00:04**	17:43	17:47	**00:04**
P2	15:38	15:50	**00:12**	15:50	16:02	**00:12**	16:02	16:15	**00:13**	16:16	16:23	**00:07**
P3	15:32	15:37	**00:05**	15:38	15:53	**00:15**	15:43	15:53	**00:10**	16:01	16:05	**00:04**
P4	15:28	15:45	**00:17**	15:45	15:54	**00:09**	15:54	16:07	**00:13**	16:09	16:13	**00:04**
P5	14:58	15:02	**00:04**	15:04	15:18	**00:14**	15:20	15:34	**00:14**	15:36	15:39	**00:03**
P6	14:36	14:43	**00:07**	14:44	15:07	**00:23**	15:08	15:19	**00:11**	15:20	15:25	**00:05**
P7	15:21	15:33	**00:12**	15:33	15:50	**00:17**	15:50	16:04	**00:14**	16:06	16:10	**00:04**
P8	15:55	16:01	**00:06**	16:01	16:17	**00:16**	16:17	16:29	**00:12**	16:30	16:35	**00:05**
P9	16:23	16:31	**00:08**	16:31	16:52	**00:21**	16:52	17:02	**00:10**	17:04	17:09	**00:05**
P10	17:31	17:38	**00:07**	17:38	17:52	**00:14**	17:52	18:04	**00:12**	18:05	18:11	**00:06**
Média			00:08			00:15			00:11			00:04

Tabela 4.1: Tempo do projeto, implementação e testes dos serviços

Os dados mostrados na Tabela 4.1 estão organizados em um único bloco que apresenta os tempos de projeto e desenvolvimento dos Web Services e testes de publicação. A Tabela 4.2 apresenta os dados referentes à quantidade total de linhas de código produzidas pelos participantes, o tempo total despendido para o projeto, desenvolvimento e testes e a produtividade de

Participante	LOC Manual	LOC Total	LOC/h	j-UDDI (t)	UDDI Semântico (t)	j-UDDI (p)	UDDI Semântico (p)
P1	52	156	425	**00:26**	**00:26**	360	360
P2	66	175	438	**00:37**	**00:31**	284	339
P3	50	146	438	**00:30**	**00:24**	292	365
P4	115	269	621	**00:39**	**00:30**	414	538
P5	76	194	647	**00:32**	**00:21**	364	554
P6	81	181	362	**00:41**	**00:35**	265	310
P7	69	174	360	**00:43**	**00:33**	243	316
P8	91	202	551	**00:34**	**00:27**	356	449
P9	103	212	439	**00:39**	**00:34**	326	374
P10	71	185	529	**00:33**	**00:27**	336	411
Média	77	189	481	00:35	00:28	324	402

Tabela 4.2: Total de linhas de código, tempo total de desenvolvimento e Produtividade

cada participante em termos de linha de código produzidas por hora. Essas informações foram calculadas apurando os dados coletados pelos participantes após a realização do experimento e os dados que foram gerados automaticamente. A linha rotulada como "Média"mostra a média de tempo gasto pelos participantes na execução de cada uma das atividades propostas, bem como o número médio de linhas de código geradas tanto manualmente quanto automaticamente. Também são mostradas as médias do total geral de linhas de código, do tempo total despendido (Mt) e da produtividade dos participantes (Mp).

Os participantes foram instruídos a relatar os problemas técnicos encontrados durante a execução do experimento. Para tanto, no formulário de coleta de dados foram colocados campos apropriados para serem preenchidos com a descrição dos problemas ocorridos, bem como o horário de identificação e de resolução dos mesmos. A Tabela 4.3 detalhas os dados preenchidos pelos participantes relacionados aos problemas enfrentados, indicando os tempos em que cada problema foi solucionado pelo participante. O tempo total gasto por cada participante para resolução dos problemas encontra-se sumarizado na coluna "Problemas"da Tabela 4.1. Esse tempo não foi descontado dos tempos totais dos participantes uma vez que problemas de ordem técnica podem ocorrer em qualquer desenvolvimento.

3) Validação dos Dados. Após a execução do experimento, verificou-se se os dados registrados pelos participantes eram coerentes, bem como se foram coletados de forma correta. Durante a apuração, constatou-se que os participantes desempenharam bem a tarefa experimental que lhes foi proposta, executando as atividades com seriedade. O treinamento realizado um

Participante	Descrição do Problema	Identificação	Resolução	Tempo Gasto
P3	Erro na importação do serviço, problema desconhecido, talvez algum caractér especial na identação do código	15:52	15:57	00:05:00
P6	Caractér com acento	15:10	15:15	00:05:00
P9	Erro na geração das ontologias, após a importação do serviço	16:41	16:51	00:10:00

Tabela 4.3: Problemas técnicos enfrentados pelos participantes

pouco antes da execução do experimento colaborou para que os participantes se habituassem ao desenvolvimento de Web Services diretamente na linguagem WSDL. Assim, durante o experimento os participantes não tiveram maiores problemas para executarem as atividades. Os tratamentos, portanto, foram aplicados corretamente conforme planejado no projeto experimental, sendo os dados coletados considerados válidos para o experimento.

4.2.4 Análise e Interpretação dos Resultados

A partir de uma análise preliminar dos dados apresentados nas Tabelas 4.1 e 4.2, alguns aspectos interessantes foram observados entre os participantes. A Figura 4.4 apresenta a distribuição média dos esforços dos participantes entre as atividades executadas no experimento.

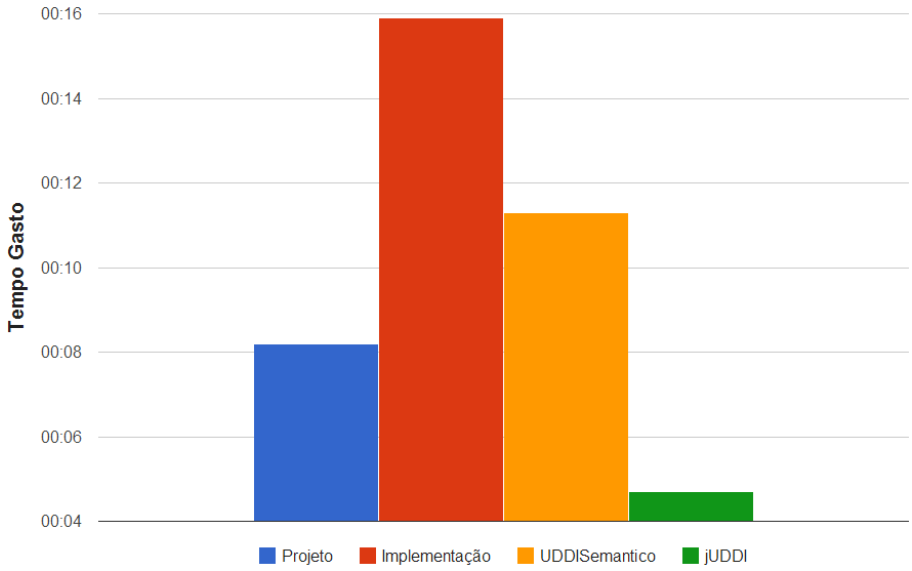


Figura 4.4: Distribuição média dos esforços por atividades

Os participantes gastaram tempos similares no projeto e implementação dos WSs. Esses tempos estão ilustrados na Figura 4.5, a qual possui as colunas Projeto, Implementação, T-UDDISemantico e T-jUDDI que representam o tempo de projeto, implementação, tempo de publicação dos serviços desenvolvidos na arquitetura proposta, utilizando UDDI Semântico

como Registro e tempo de publicação dos serviços desenvolvidos na arquitetura padrão, utilizando jUDDI como Registro, respectivamente.

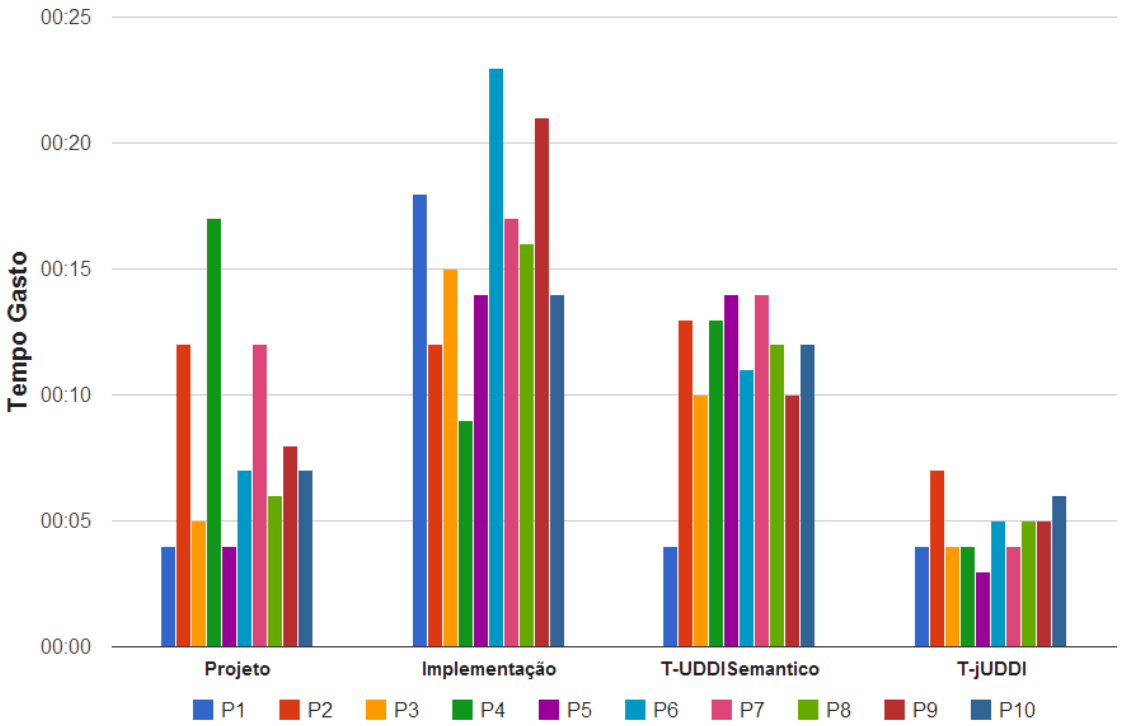


Figura 4.5: Distribuição dos esforços das atividades por participantes

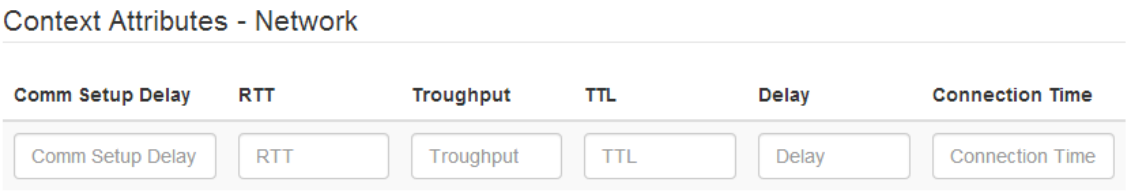


Figura 4.6: Formulário de coleta de informações contextuais do serviço

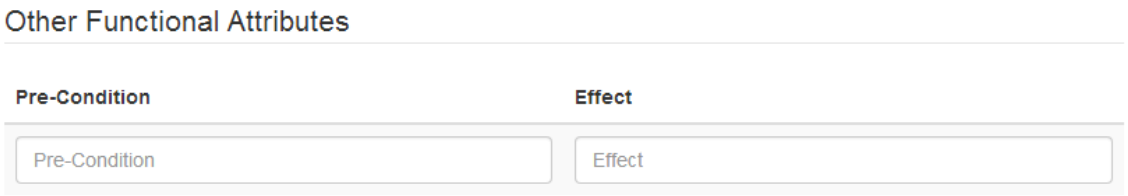


Figura 4.7: Formulário de coleta de informações relacionadas a pré-condições e efeitos do serviço

Essa relação, inversamente proporcional entre tempo despendido e quantidade de linhas de código geradas, pode ser justificada pelo emprego das transformações do WSDL em OWL-s e a criação das ontologias OWL com informações contextuais da rede. Os dados evidenciam, através do tempo médio total, que o esforço de implementação pode ser significativamente reduzido sem afetar negativamente a produtividade da equipe, uma vez que grande parte das

tarefas de codificação podem ser encapsuladas nas transformações e gerações de código, sendo que para utilização de algum serviço necessita-se descobri-los. Ao mesmo tempo, o esforço de projeto e implementação são os mesmos, pois o UDDI Semântico, componente da arquitetura proposta, foi projetado para suportar as diferentes versões de WSDL. Deve-se ressaltar que foram publicados serviços descritos em RESTful e SOAP no UDDI Semântico, mas apenas serviços baseados em SOAP no repositório jUDDI.

Após essas observações iniciais, passou-se para os passos de obtenção de conclusões do estudo de maneira estatística, conforme descrito a seguir:

O objetivo dessa etapa do experimento foi verificar, com algum grau de significância, se é possível rejeitar a hipótese nula (H_0) em favor de alguma das hipóteses alternativas (H_1 ou H_2) com base no conjunto de dados obtido.

Dado que a publicação dos WSs foi considerada apenas na etapa de testes, já tendo estes sido projetados e implementados, pôde-se desmembrar o conjunto de dados obtidos em dois subconjuntos distintos, cada qual considerando o tempo total gasto pelos participantes (**t**) e suas respectivas produtividades (**p**) em relação à adoção de cada uma das arquiteturas. A Tabela 4.4 resume esses dados.

Participante	Projeto	Implementação	Teste j-UDDI	Teste UDDI-Semântico	LOC	(t) UDDI - Semântico	(t) j-UDDI	(p) UDDI-Semântico	(p) j-UDDI
P1	**00:04**	**00:18**	**00:04**	**00:04**	353	**00:26**	**00:26**	815	815
P2	**00:12**	**00:12**	**00:13**	**00:07**	175	**00:37**	**00:31**	284	339
P3	**00:05**	**00:15**	**00:10**	**00:04**	146	**00:30**	**00:24**	292	365
P4	**00:17**	**00:09**	**00:13**	**00:04**	269	**00:39**	**00:30**	414	538
P5	**00:04**	**00:14**	**00:14**	**00:03**	194	**00:32**	**00:21**	364	554
P6	**00:07**	**00:23**	**00:11**	**00:05**	181	**00:41**	**00:35**	265	310
P7	**00:12**	**00:17**	**00:14**	**00:04**	174	**00:43**	**00:33**	243	316
P8	**00:06**	**00:16**	**00:12**	**00:05**	202	**00:34**	**00:27**	356	449
P9	**00:08**	**00:21**	**00:10**	**00:05**	212	**00:39**	**00:34**	326	374
P10	**00:07**	**00:14**	**00:12**	**00:06**	185	**00:33**	**00:27**	336	411
Média						00:35	00:28	369	447

Tabela 4.4: Resumo dos dados coletados

A princípio, analisando os dados obtidos no experimento, aparentemente o uso da arquitetura proposta foi suplantado pelo uso da arquitetura padrão com o jUDDI, o que nos induz a validar a hipótese alternativa H_2 em detrimento de H_1. Contudo, conforme descrito anteriormente, essa defasagem é posteriormente compensada em decorrência da descoberta automática de serviços advinda pela agregação de significado nos WSs publicados no UDDI Semântico.

Para verificar esse efeito estatisticamente, aplicou-se o teste chamado t-test (MONTGOMERY, 2008). Esse teste paramétrico é usado para comparar duas amostras independentes e

checar se as médias de seus dados são estatisticamente diferentes e, dessa maneira, mostrar que o efeito hipotético foi demonstrado. No caso do estudo realizado, as amostras foram constituídas pelos dados referentes aos tempos totais e produtividades tanto do uso da arquitetura proposta, quanto do uso da arquitetura padrão.

Uma vez que o efeito na variável dependente do experimento (eficiência dos participantes) envolveu dois aspectos, tempo total (**t**) e produtividade (**p**), a aplicação do t-test ao conjunto amostral de dados foi realizada em duas etapas. Na primeira etapa, comparou-se as amostras relativas aos tempos totais de cada participante em relação ao uso das duas arquiteturas. De maneira análoga, na segunda etapa, a comparação foi feita com as amostras referentes às produtividades. O teste da hipótese nula se baseou na combinação dos critérios de rejeição das duas etapas do teste. Por exemplo se as hipóteses forem rejeitadas então a hipótese nula é aceita.

Primeira etapa do teste: H_0t: $Mt_{proposta} = Mt_{padrao}$

Aplicando o *t-test* sobre as amostras referentes ao tempo total decorrente do uso de ambas as arquiteturas, foi possível rejeitar a hipótese nula que declara igualdade entre as médias de tempo total gasto pelos participantes com $p < 0{,}05$ ($p = 0{,}00805823$).

Segunda etapa do teste: H_0p: $Mp_{proposta} = Mp_{padrao}$

Aplicando o *t-test* nas amostras referentes às produtividades, foi possível rejeitar a hipótese nula que declara que não há diferenças entre as médias de produtividade dos participantes usando ambas as arquiteturas com $p < 0{,}05$ ($p = 0{,}000983536$).

Uma vez que se pôde rejeitar a hipótese H_0 nas duas etapas do testes, e tendo em vista que $Mt_{proposta} > Mt_{padrao}$ e $Mp_{proposta} < Mp_{padrao}$, pode-se validar a hipótese alternativa H_1 em detrimento de H_2.

4.2.5 Ameaças à Validade

Uma das questões fundamentais envolvendo os resultados de um experimento é determinar quão válidos eles são. Por isso, é importante considerar a questão da validade da avaliação desde a fase de planejamento de forma a guiar a execução do experimento no sentido de obter uma validade adequada dos resultados. Validade adequada, neste caso, significa que os resultados são válidos para a população de interesse, tanto para o subconjunto populacional que executou o estudo, quanto para uma população mais ampla para a qual os resultados podem ser generalizados (WOHLIN et al., 2000).

As principais ameaças que podem colocar em risco a validade de um experimento se classi-

ficam que quatro tipos (WOHLIN et al., 2000): de conclusão, interna, de construção e externa. Essas ameaças foram tratadas no experimento realizados, conforme segue:

1) Validade de Conclusão. Refere-se às questões que afetam a habilidade de tirar conclusões corretas a respeito do efeito dos tratamentos nas variáveis (e.g. cuidados tomados na execução e na medição do experimento). As medidas que deveriam ser coletadas pelos participantes envolveram dados diretos (hora, LOC) que independem de julgamento humano, ao contrário de medidas como *pontos por função* (PRESSMAN, 2006), por exemplo. Isso aumentou a confiabilidade dos dados coletados, mesmo sob o risco de possíveis imprecisões nos dados, já que estes foram coletados pelos próprios participantes. Além disso, a heterogeneidade da experiência dos participantes do experimento pode afetar os resultados, apesar de estarem próximas.

2) Validade Interna. Refere-se às questões que afetam a habilidade de assegurar que os resultados foram, de fato, obtidos em decorrência dos tratamentos e não por uma coincidência, ou pelo efeito de outro fator que não foi medido ou não se pôde controlar (e.g. modo como os participantes são selecionados, tratados; situação em que ocorre o experimento). No caso deste estudo, buscou-se realizar o experimento com estudantes de pós-graduacão da área de Computação, os quais já possuíam certo nível de experiência com desenvolvimento de software, conforme reportado em seus formulários de caracterização. Assim, assumiu-se que eles são representativos para a população dos desenvolvedores de software. Ainda, o estudo foi realizado em duas sessões, a primeira com 5 participantes, assim como a segunda.

3) Validade de Construção. Refere-se às questões que afetam a habilidade de generalizar o resultado do experimento ao conceito ou teoria envolvidos no estudo (e.g. definição adequada das medidas e tratamentos). No estudo realizado, o objetivo foi comparar dois repositórios de serviços com respeito ao impacto de eficiência na publicação e descoberta de serviços. Para os propósitos do estudo, um participante eficiente é aquele que produz mais código e finaliza a implementação em menos tempo. Portanto, os dados para coleta e os tratamentos foram definidos de modo que fosse possível realizar adequadamente a análise do efeito na eficiência das equipes em conformidade com os objetivos do estudo. Além disso, a fim de evitar interações dos participantes voltadas para maximização ou minimização das métricas de interesse (produtividade e tempo despendido), evitou-se divulgar quais métricas seriam analisadas quando os mesmos foram informados dos objetivos. Isso também evitou que os participantes se comportassem de maneira diferente do comum, buscando se empenhar melhor ou agir de forma negligente mais do que de costume apenas para obterem resultados que favorecessem ou prejudicassem o experimento. O tratamento dessa ameaça permitiu reduzir o impacto de dados inválidos para o

estudo.

4) Validade Externa. Refere-se às questões que afetam a habilidade de generalizar os resultados do experimento para um contexto mais amplo do que foi selecionado para o estudo. Neste casos, existem três riscos principais: a escolha errada dos participantes, conduzir o experimento em um ambiente inapropriado, e desempenhar o estudo, em geral, podem ser considerados representativos para a população dos desenvolvedores de software. O experimento foi conduzido em um laboratório informatizado com equipamentos adequados, bem como com ferramentas e tecnologias de desenvolvimento atualizadas comumente empregadas em ambientes industriais. Com relação às questões temporais, o experimento foi planejado de forma que os estudantes pudessem desempenhá-lo dentro de um período de 1 hora, evitando que os resultados fosse afetados em decorrência de tédio ou desgaste excessivo dos participantes.

4.3 Considerações Finais

Este capítulo apresentou a avaliação do trabalho desenvolvido. A avaliação compreendeu um estudo de caso, em que investigou-se a aplicabilidade do UDDI Semântico e o comportamento dos serviços e suas ontologias, assim como o estudo de um experimento realizado com a intenção de melhor avaliar a arquitetura de acordo com testes com usuários. Os resultados favoráveis obtidos com a avaliação do trabalho vão ao encontro das expectativas iniciais do projeto, que incluem redução dos esforços na descoberta de serviços, aumento da produtividade e da qualidade na descoberta dos serviços.

Capítulo 5

TRABALHOS CORRELATOS

Diversos trabalhos relacionados ao desenvolvimento de aplicações sensíveis ao contexto e descoberta de serviços tem sido propostos pela comunidade acadêmica, o que demonstra a relevância desse tema de pesquisa. Este capítulo apresenta alguns destes trabalhos que possuem certa relação com o trabalho desenvolvido nesta pesquisa.

A organização do capítulo está conforme segue: a Seção 5.1 apresenta os principais trabalhos encontrados na literatura; a Seção 5.2 fornece uma análise comparativa dos trabalhos correlatos ao trabalho aqui desenvolvido; por fim, a Seção 5.3 conclui o capítulo tecendo algumas considerações finais.

5.1 Trabalhos Encontrados na Literatura

Algumas soluções pesquisadas na literatura objetivam a definição de frameworks que disponibilizam meios para adicionar semântica em serviços Web do tipo RESTful (JOHN; RAJASREE, 2012). Outros definem soluções com a inclusão de ontologias para seleção baseada em qualidade de serviço (QoS) (NAKAMURA, 2012). Há também pesquisas relacionadas a personalização de serviços em diferentes contextos de aplicação (TEGEGNE; KANAGWA; WEIDE, 2010b; TEGEGNE; WEIDE, 2011).

5.1.1 A framework for the Description, Discovery and Composition of RESTful Semantic Web Services

O trabalho de John e Rajasree (2012) é uma abordagem proposta para apoiar o desenvolvimento de serviços web semânticos RESTful, devido ao fato desse tipo de arquitetura ser mais simples – por funcionar ao longo do protocolo HTTP – que a RPC dos serviços SOAP. John e

Rajasree (2012) tratam o problema que as soluções propostas atualmente são extensões de soluções aplicáveis para serviços SOAP e que requerem arquivos de descrição externa deixando os desenvolvedores com mais um artefato para desenvolver ou manter. Com base nesse problema eles criaram um framework que possibilita a adição de semântica em serviços web RESTful.

A solução proposta utiliza uma combinação de marcação de estilo *Microformats* e RDFS para fornecer um framework abrangente que possibilite descrever, descobrir e compor serviços RESTful por adição da semântica. *Microformats*, sendo simples e de fácil reutilização das propriedades do HTML, oferece aos usuários uma entrada baixa resistência dos desenvolvedores, o que pode aumentar a taxa de adoção.

Essas anotações não são visíveis para o usuário, mas ficam escondidas em tags do HTML, portanto, não entram no caminho de usuários navegando pelo site. A fim de permitir forte interligação entre os serviços, é necessária uma solução mais robusta como RDF e RDF Schema, para certificar as propriedades do framework e isto é alcançado através do fornecimento de um *Adapter* para a conversão automática dos Microformats para RDF e RDF Schemas, proporcionando, desta forma, aos desenvolvedores uma forma automática de conversão que funcionam em segundo plano.

Em geral a descrição do serviço é feita diretamente no HTML, utilizando anotações no próprio código, sendo acessados por JavaScript ou CSS. O mecanismo de descoberta trabalha para habilitar a descoberta de serviços novos, similares e afins, abordando dois tipos de descobertas: descoberta pelos usuários e descoberta pelos serviços. A descoberta pode ser feita diretamente pelo browser (com uma extensão apropriada, desenvolvida pelo próprio autor), indicando outros serviços REST naquele mesmo domínio, como em um leitor de RSS.

A descoberta automática permite encontrar serviços similares dentro do mesmo domínio da aplicação, comparando seus atributos e formando grafos com serviços similares. Outro tipo de descoberta é comparando os serviços que são gerados a partir de um serviço em comum, considerado *Pai*, com outros considerados *Filhos* pressupondo que eles poderão ser similares. E o último é unir os clientes e consumidores procurando os serviços por quem forem consumidos considerando que os consumidos-por serão similares.

Sobre a composição os autores utilizam as mesmas anotações da descoberta com ênfase em composição, construindo um grafo, o qual é percorrido, possibilitando uma composição. John e Rajasree (2012) fizeram um estudo de caso com serviços desenvolvidos em PHP que simulam uma livraria, um catálogo de serviços e uma loja de livros, com alguns serviços similares e obtiveram resultados significantes para sua pesquisa. Também testaram o plugin que funciona somente com o navegador Google Chrome.

Neste trabalho foi concluído que adicionar semântica para descrição de serviços é um passo importante para uma web melhor, acessível tanto para seres humanos quanto para máquinas. A solução proposta alinha com a filosofia e arquitetura REST e reutilizações de documentações de serviços existente para transforma-los em descrições legíveis por máquina. É uma solução simples, fácil de escrever e manter na perspectiva de um desenvolvedor, mas ainda precisa ser melhorada abordando também serviços SOAP.

5.1.2 DaaS: Cloud-based mobile Web Service discovery

Elgazzar, Hassanein e Martin (2013) mostra uma preocupação em relação a proliferação de *smartphones* e o acesso sem fio já onipresente entre as pessoas atualmente e a relação com serviços web e a descoberta de serviços relevantes que se encaixam no contexto de utilização destas pessoas, e afirma que, com o avanço na computação em nuvem se torna viável realizar a descoberta de serviços móvel (para dispositivos móveis).

Eles apresentam alguns tipos de descobertas de serviços, baseadas em UDDI, motores de busca especializados, motores de busca genéricos da web e as limitações destes no contexto da computação para dispositivos móveis. Baseando-se nestas limitações criaram o **DaaS** - Discovery as a Service, um novo framework de descoberta baseado na computação em núvem que aborda os principais componentes de descoberta de serviços móvel. Este framework, estabelece o fundamento da descoberta eficiente de serviços web para dispositivos móveis levando em consideração o contexto e as preferências do usuário.

O DaaS tem dois objetivos principais: (1) realizar todo o processamento intensivo de descoberta de serviços utilizando os recursos da nuvem, a fim de poupar os escassos recursos móveis; (2) incorporar as preferências do usuário e seu contexto na descoberta de serviços web móvel. O framework é dividido em três camadas:- *Client*, *Cloud* e *Provider*.

O funcionamento do framework é baseado na requisição (*Service Request*) de um usuário, que ocorre na camada *Client* da aplicação. O usuário submete uma descrição em "texto plano"com seus objetivos. O "texto plano"é utilizado devido a limitação de recursos dos dispositivos. Uma análise mais aprofundada dos dados de entrada é realizada pelo *Request Handler* que extrai as palavras-chave e informações significativas e utiliza uma combinação com base nas palavras-chave.

O *Context Manager* é responsável por coletar o contexto do ambiente que se encontra o usuário utilizando sensores embarcados através das capacidades do dispositivo móvel, assim como suas preferências. Tal informação de contexto é usada para classificar os serviços rele-

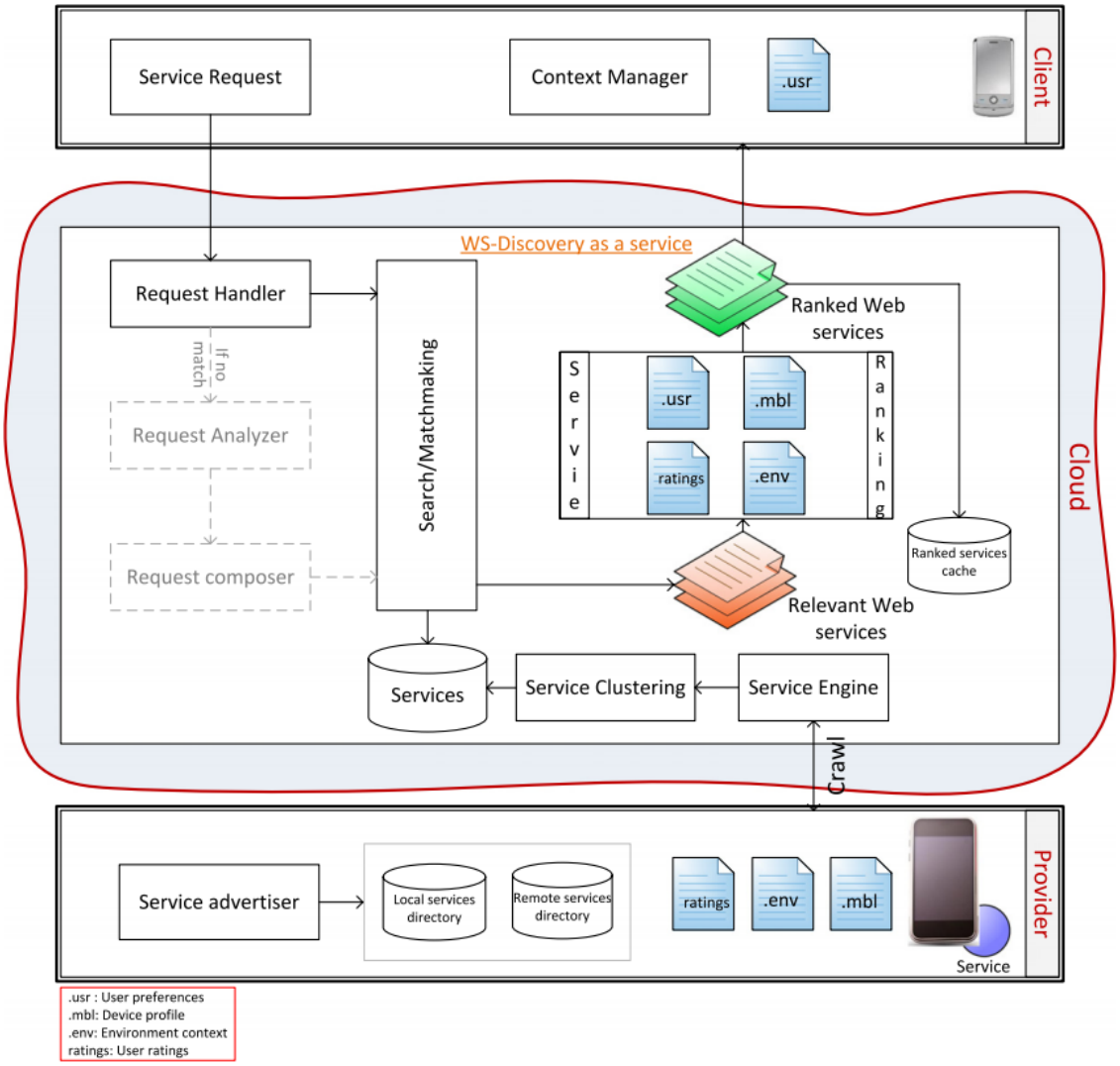


Figura 5.1: Visão geral do framework DaaS (ELGAZZAR; HASSANEIN; MARTIN, 2013).

vantes de acordo com os que se encaixam melhor em tal contexto. *Context Manager* também monitora as redes disponíveis e seus status.

A camada *Cloud* contem os componentes a serem executados para processar a requisição realizada na camada de cliente. Nesse camada ocorrem operações de processamento avançadas sobre a solicitação do usuário, extraindo as palavras-chave utilizando técnicas de análise de texto. Primeiramente é realizada uma comparação para averiguar se a primeira busca é realizada com sucesso, se não a requisição é analisada e as palavras são utilizadas para executar analisador da requisição, logo após formatar a solicitação de serviço que podem combinar com outros tipos de serviço ou diferentes linguagens de descrição (WSDL, OWL-S, WSM, WSDL-S, etc.) de acordo com a disponibilidade destes serviços no repositório e realizar a busca novamente.

A camada *Provider* contém os componentes que são implementados no lado do provedor. No cenários de serviços móveis, fornecedores de serviços podem ser uma entidade móvel com

recursos limitados. O framework DaaS remove a carga sobre estes provedores de recursos limitados, deslocando processos intensivos para a nuvem, mantendo apenas os componentes necessários nos dispositivos móveis. Portanto, neste contexto, o único componente que os provedores precisam é o módulo responsável por anunciar a disponibilidade do serviço. *Service advertiser* é o responsável por anunciar estes serviços publicando seus serviços localmente, similar ao publicar em um UDDI. Cada dispositivo gerencia o que quer/pode oferecer tanto localmente quanto remotamente. O diretório que estão publicados os serviços representam os descritores de serviço que expressam suas funcionalidades como titulo, descrição, URI local, *endpoint*, etc. Esses são os parâmetros utilizados nas comparações da descoberta.

Para a avaliação foi utilizado um ambiente de nuvem da amazon EC2, com diversos representações de contextos, serviços e usuários. Foram avaliados tempo de resposta, impacto de clusterização, sobrecarga de contexto, escalabilidade e precisão e qualidade da descoberta.

A validação experimental e avaliação de desempenho demonstram que a DaaS pode classificar de forma eficaz os serviços relevantes de acordo com o contexto do usuário e suas várias preferências, além de melhorar a precisão dos serviços descobertos. O protótipo de software desenvolvido também mostra que o agrupamento de serviços web para a descoberta melhora significativamente o tempo de resposta global, enquanto a nuvem mantém a escalabilidade de acordo com critérios de desempenho pré-especificados.

5.1.3 Utilização de Web Semântica para seleção de informações de Web Services no registro UDDI: Uma Abordagem com qualidade de serviço

Na pesquisa de Nakamura (2012), para obtenção de título de mestre, foi criada uma ontologia com propósito de descoberta universal com ontologia e QoS e um módulo de sistema que tem a função de pesquisar serviços no registro de serviços (UDDI), baseando-se em parâmetros de QoS. Este módulo faz uso de Web Semântica e da ontologia criada por ele. As informações sobre os acordos entre clientes e provedores, bem como a qualidade de serviço e seus atributos estão em uma base de conhecimento que está representada pela ontologia UDOnt-Q (*Universal Discovery with Ontology and QoS*), nomeada assim pelo autor.

Entre as diversas possibilidades de acesso que poderiam ser utilizadas para acessar o módulo de descoberta, optou-se por transformar o módulo em um *Web Service* (WS) para garantir a interoperabilidade de acesso pelos mais diversos clientes (NAKAMURA, 2012). Neste WS existe um método, GetWSDL, no qual é passado o nome do serviço buscado. A Figura 5.2 ilustra a sequencia de execução do módulo desenvolvido, após uma breve explicação sobre esse

fluxo.

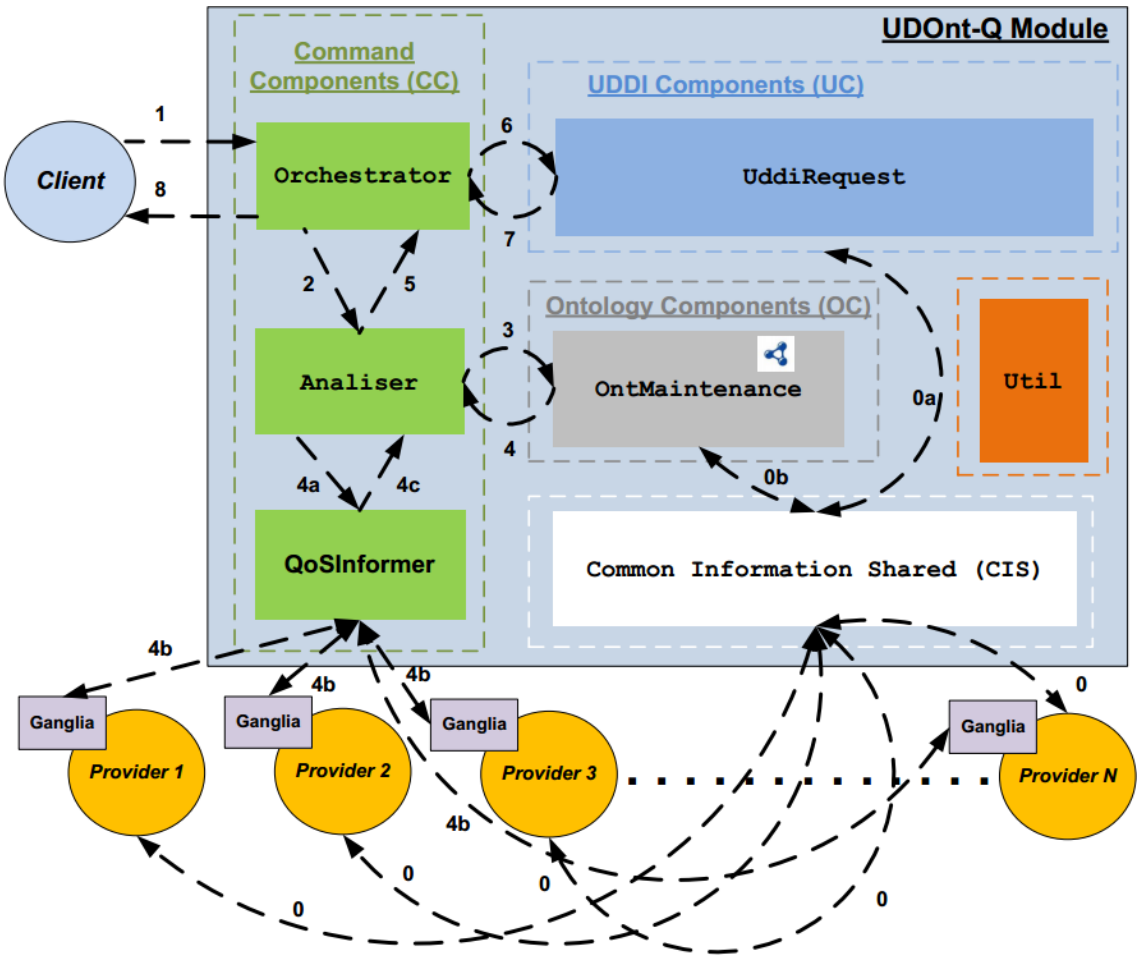


Figura 5.2: Estrutura resumida do Módulo com os seus componentes (NAKAMURA, 2012)

0. Os provedores implementam a classe **CIS**, após registram os seus serviços no UDDI(0a) e na ontologia(0b);

1. O cliente acessa o WS e solicita um serviço informando o nome do serviço;

2. O componente **CC** encaminha a requisição para o Analiser, a fim de verificar qual componente de busca será utilizado. Nesse trabalho somente 1 componente de busca foi desenvolvido, no caso o **OC**;

3. O componente **OC** recebe as informações e instancia o algoritmo de busca por Objeto ou SPARQL, definido na configuração do módulo. Dessa forma, baseando-se no acordo determina-se a QoS contratada pelo cliente e também quais serviços atendem aos requisitos dessa QoS. O resultado da busca é um conjunto de dados composto pelo nome do serviço, o nome do provedor e o endereço do provedor.

4. O **CC** recebe e analisa o resultado da busca feita pelo algoritmo. A análise consiste em verificar se há um ou mais serviços que atendam à requisição do cliente. Existindo apenas um, o serviço é retornado à classe *Orchestrator* e o sistema continuará o fluxo. Caso haja mais de um serviço, o sistema pode seguir dois fluxos que dependem da opção de monitoramento indicada na configuração, a qual pode estar habilitada e encaminhar a lista com os resultados para o **QC** ou estar desabilitada retornando o primeiro resultado da lista;

5. A classe *Orchestrator* recebe o resultado, recupera o nome do serviço e do provedor e encaminha para o componente **UC**;

6. O componente **UC** procede com a pesquisa no registro UDDI, procurando a o WSDL do serviço daquele determinado provedor;

7. O arquivo WSDL é retornado ao componente **CC** que a repassa ao cliente;

8. O cliente recebe o arquivo WSDL do serviço que possui a qualidade contratada e a partir dela é possível determinar a localização do serviço e posteriormente utilizá-lo. Caso ocorra algum erro ou nenhum serviço seja encontrado é retornado o valor *Null* ao cliente.

Este trabalho apresentou contribuições para descoberta de Web Services baseada em QoS, com uso de ontologias, dentro do contexto da SOA, considerando a seleção de algumas informações de qualidade de serviço de WSs, afim de classifica-los por meio de recursos da Web Semântica, utilizando motores de inferência, sobre as ontologias, utilizadas através de informações do contrato entre cliente e servidor. Dentre algumas limitações desse trabalho as principais são:- busca dos serviços acontece somente com palavras-chave, dificultando uma descoberta mais detalhada; apenas serviços baseados em SOAP são utilizados, sendo que no cenário atual, serviços baseados em REST estão em grande ascensão.

5.1.4 eHealth Service Discovery Framework for a Low Infrastructure Context

Tegegne, Kanagwa e Weide (2010a) fizeram um trabalho muito importante em um contexto de baixa infra-estrutura em saúde, aplicando conhecimentos de computação e saúde. Com esse trabalho tentaram resolver o problema, das informações contidas nos registros dos pacientes não serem utilizadas para fornecer acesso rápido e personalizado para os pacientes de serviços de saúde, muito menos um tratamento adequado, especialmente em um contexto de baixa infra-estrutura, o qual os prestadores de serviços de saúde são muitas vezes dominados por grandes

números que levam à degradação na prestação de serviços. O foco deste trabalho foi criar um framework para descoberta serviço *eHealth* no contexto de baixa infra-estrutura, visando a descoberta de serviços específicos em um domínio, com personalização do mesmo, proporcionando serviços melhores para os paciente. Para fazer isso, Tegegne, Kanagwa e Weide (2010a) categorizaram o contextos de acordo com diferentes perfis dos usuários. O framework oferece ontologia baseada em semântica sensível ao contexto e serviços personalizados.

Baseado nesse contexto Tegegne, Kanagwa e Weide (2010a) propõem uma arquitetura ilustrada na Figura 5.3. Os autores apresentam um exemplo que ilustra a ação do framework e o fluxo de ações que é executado. Primeiramente o HEW (Health Extension Workers) – pessoa que atende o paciente – passa dados de contexto do paciente para aplicação, logo a aplicação identifica os dados de perfil e contexto do paciente e também informações além das passadas pelo HEW, como temperatura, clima, distancia do hospital mais próximo e retorna ao HEW. Logo após a aplicação interpreta a situação do paciente, e compara com as situações do repositório, oferecendo alguns diagnósticos, prevenções e medicamentos possíveis ao HEW, auxiliando no diagnóstico da doença do paciente.

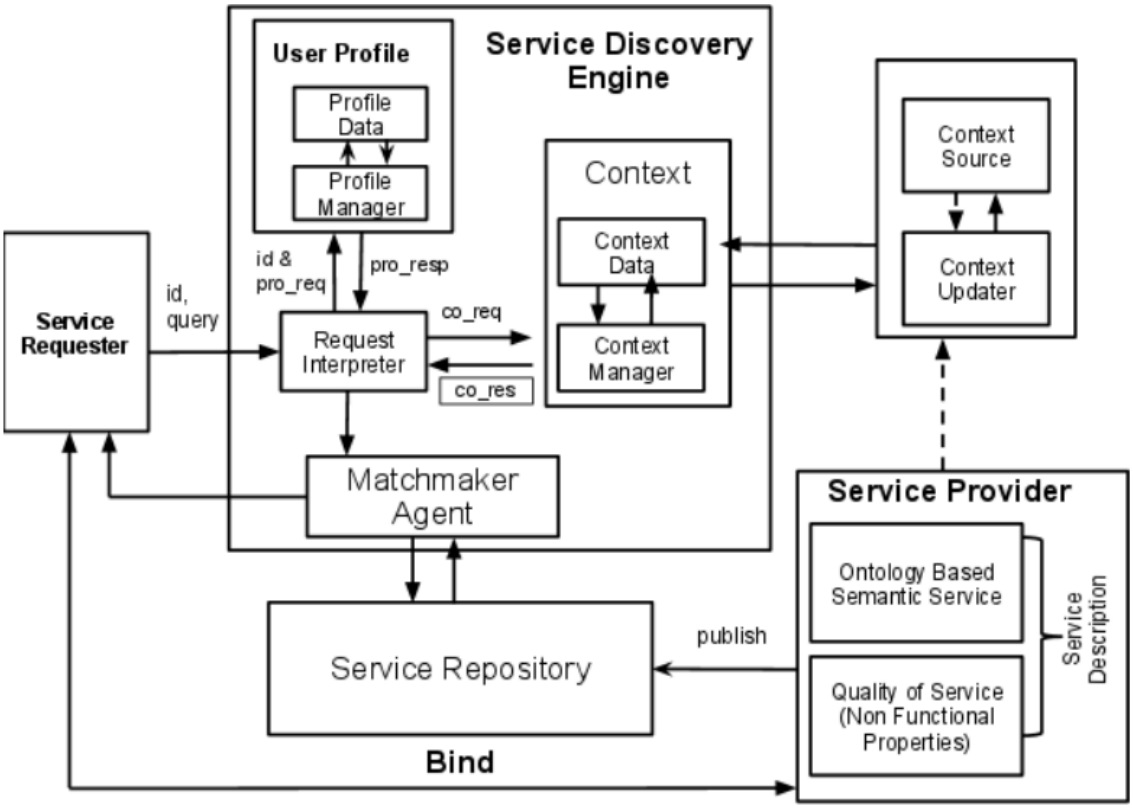


Figura 5.3: Framework de Descoberta de Serviços (TEGEGNE; KANAGWA; WEIDE, 2010a)

Na fase de descoberta semântica de serviços, a seleção de serviços envolve um *Matching* semântico e *Ranking* para selecionar um único serviço mais relevante a ser invocado, começando

a partir de um determinado conjunto de serviços disponíveis. O *Matching* é a comparação par a par de um serviço anunciado com um serviço desejado para determinar o grau de sua correspondência semântica. Este processo pode ser não baseado em lógica, baseada em lógica ou híbrido, dependendo sobre a natureza do estilo raciocínio utilizado pelo *Matcher* para calcular correspondência parcial ou totalmente, ordenando por graus entre representações de semântica de serviços. Classificação subsequente de serviços determina a ordem de seus graus individuais de correspondência semântica com a consulta dada.

O agente atualizador de contexto fornece informações adicionais sobre um serviço e um usuário, por exemplo, a localização. Para esse efeito, o contexto atualizado está equipado com agentes especiais. Por exemplo, um agente de GPS que pode controlar a posição do usuário, que tem a vantagem adicional de se utilizar informação relacionada com a localização. Esta informação extra irá melhorar a descoberta de serviços. O contexto pode variar com o tempo, por isso a atualização contexto pode também envolver uma função de rastreamento para sinalizar quando o contexto muda. O atualizador contexto pode ser um agente digital na Internet, um humano ou um sistema. Durante a descoberta de serviço o contexto do usuário é comparado com o contexto do serviço, a fim de recuperar os serviços relevantes no que diz respeito a ciência do contexto. O contexto é muito importante para a descoberta de serviços em uma rede de infra-estrutura baixa, uma vez que ele pode recuperar os serviços que estejam de acordo com o contexto do serviço atual do usuário. Os serviços de saúde são muito sensíveis, uma vez que os serviços são relacionados para evitar mortes de seres humanos. Por isso, o contexto do paciente (tais como pressão sanguínea, a temperatura e outros sintomas) deve ser precisamente fornecidos remotamente para uma prescrição adequada, de consultadoria e apoio. Caso contrário, seria complicado para tratar o paciente e prestar assistência ao profissional de saúde a distância.

O framework contém um motor de descoberta de serviços, baseado em dados de contexto, gerenciador de contexto, dados de perfil do paciente, gerenciador de perfis um interpretador de requisições, um *Matchmaker Agent* e um repositório de serviços.

Os dados de contexto contem o contexto dos serviços e dos usuários. O contexto da solicitação será fornecido pelo fornecedor de contexto. Dois tipos de contexto são armazenadas no banco de dados de contexto: contexto do serviço e do usuário contexto. Contexto Serviço inclui informações sobre serviços, tais como localização, versão, custo e identidade do provedor. Contexto do usuário, um contexto diferente do usuário perfil, contém informações como localização, clima, condição, o status do usuário (ocupado, no telefone), o tempo.

O gerenciador de contexto entra em contato com o dados de contexto do repositório para atualizar ou modificar o contexto do utilizador. Por exemplo, o atualizador contexto pode des-

cobrir o usuário mudou, e, assim, enviar uma atualização solicitar ao gerente de contexto. Ele também é responsável por agregar informações de contexto decorrentes ou novas com base em regras específicas de domínio.

Os dados do perfil do usuário incluem dados biográficos pessoais, preferências, regras e restrições. Isso pode ser atualizado com base no comportamento do usuários, por exemplo, usando o serviços invocados ou a história de uso. As preferências do usuário proporcionam um mecanismo de personalização, o que pode ser visto como parte do contexto global, e habilitar o serviço descoberta de uma forma que corresponda melhor aos requisitos do usuário implicitamente ou explicitamente. O perfil de usuário deve ser armazenado de forma segura com base no nível de privacidade que o usuário concordou.

O gerenciador de perfil recebe o perfil do gerenciador de contexto. Logo após ele receber o perfil do solicitante/consumidor verifica se o solicitante tem um perfil na base de dados de perfis. Se o solicitante/consumidor é novo o sistema irá enviar o perfil para o base de dados. Depois de verificar o perfil do usuário, preferência, contexto, o intérprete perfil envia os dados para o gerenciador contexto. O gerenciador de contexto irá adicionar as informações do perfil com as informações de contexto que será enviado para o agente de *matchmaking*.

O interpretador de requisições recebe o pedido do consumidor e propaga a consulta no gerenciador de contexto com o perfil. Este dois agentes extraem a informação a partir da consulta e verificam as informações comparando-as com as informações no banco de dados. O gerenciador de contexto depois de receber a resposta do gerenciador de perfil, que irá transmiti-las ao agente de *matchmaking* para combinar com o pedido contra o especificação do serviço no registro de serviço.

Matchmaker Agent de modo geral dividido em dois: *matchmaking* sintática que utiliza a estrutura ou formato de serviço solicitante com o serviço do servidor. A segunda utiliza a semântica ou significado do serviço do solicitante com o serviço do provedor. Este agente *matchmaking* vai reunir serviços solicitados a partir do interpretador de consulta e o serviço semântico baseado em ontologia dos serviços no Registro. Com base nestas informações, o *matchmaking* agente decide se o pedido corresponde ou não ao serviço solicitado.

Este trabalho não aborda tecnologias utilizadas, apresenta apenas a teoria da aplicação, que é testada em outro trabalho do mesmo autor, que retrata o funcionamento do framework (TEGEGNE; KANAGWA; WEIDE, 2010b; TEGEGNE; WEIDE, 2011). Diante disso, este trabalho, propõe um framework de descoberta de serviços de saúde que facilita a eficácia do *eHealth*. Está inserido no contexto de ciência do contexto e SOA, por isso foi considerado um correlato a está pesquisa. A estrutura oferece diversas facilidades para criar, especificar,

descobrir e selecionar os serviços de saúde. Especialmente, o framework cobre a parte ignorada da orientação a serviços para personalizar os serviços com base nas necessidades do paciente. Esta estrutura serve tanto para redes com fio e como para redes sem fio.

5.2 Análise Comparativa

O trabalho desenvolvido nesta pesquisa se baseia em diversas características dos trabalhos anteriormente descritos. Contudo, apresenta suas próprias contribuições através da evolução e/ou adequação das concepções dos trabalhos correlatos.

Em geral, os trabalhos relacionados diferem-se do trabalho desta pesquisa em relação ao uso de dois tipos de tecnologias arquiteturais para implementação dos Web Services, que são: SOAP e RESTful. Também é um diferencial deste trabalho o uso de tecnologias que possibilitam a sensibilidade ao contexto computacional, que possibilita o desenvolvimento de aplicações sensíveis ao contexto, sendo auxiliado pelo uso das ontologias dos serviços.

John e Rajasree (2012), por exemplo, propõe um framework para apoiar o desenvolvimento de serviços web semânticos RESTful, tratando o problema que as soluções propostas atualmente são extensões de soluções aplicáveis para serviços SOAP e que requerem arquivos de descrição externa deixando os desenvolvedores com mais um artefato para desenvolver ou manter. Fazendo uma comparação com o trabalho proposto, esse framework aborda apenas serviços RESTful, e não utiliza sensibilidade de contexto para descoberta dos serviços.

O framework desenvolvido por Elgazzar, Hassanein e Martin (2013) apresenta um novo termo na computação: **DaaS** - Discovery as a Service - Descoberta como serviço, o qual, dentro dos limites da computação móvel, tem o objetivo de descobrir serviços entre dispositivos móveis, baseado na computação em nuvem, fugindo do modelo de UDDI que é utilizado neste projeto. Mas, no aspecto de ciência de contexto, cumpre os requisitos de contexto computacional e de preferências de usuário.

Nakamura (2012) criou um módulo para descoberta de serviços, baseado na ontologia UDOnt-Q, de sua autoria, e em atributos de QoS, com propósito de descoberta universal de serviços em um UDDI. O módulo faz uso de Web Semântica e da ontologia criada por ele, agregando apenas serviços baseados em SOAP. As informações sobre os acordos entre clientes e provedores, bem como a qualidade de serviço e seus atributos estão em uma base de conhecimento que está representada pela ontologia UDOnt-Q. Nakamura (2012) representa os atributos de QoS em sua própria ontologia, assim como os atributos dos descritores de serviço utilizados, diferentemente do utilizado neste trabalho, o qual utiliza o framework UbiCon Estendido

para gerenciamento do contexto de rede, que também engloba atributos de QoS, separando os arquivos de contexto, de descrição semântica e sintática, facilitando o reúso de cada arquivo separadamente.

Tegegne, Kanagwa e Weide (2010a) criaram um framework com a proposta de descoberta de serviços, específicos para o domínio da saúde, baseada no contexto do usuário, com personalização do mesmo, proporcionando serviços melhores para os paciente. Eles propõem uma arquitetura utilizando ciência de contexto e serviços para resolver o problema de saúde de um país da Africa, fornecendo acesso rápido e personalizado para os pacientes. O framework oferece ontologia baseada em semântica sensível ao contexto e serviços personalizados, como principal contribuição. Com base na arquitetura deste sistema, foi criado o modelo arquitetural deste projeto, que também utiliza a ontologia OWL-S para descoberta de serviços e o framework UbiCon Estendido para gerenciar o contexto, assim trabalhando com sensibilidade ao contexto.

A Tabela 5.1 resume as principais características analisadas entre os trabalhos.

Características	**Trabalhos**				
	Proposta	A framework for the Description, Discovery and Composition of RESTful Semantic Web Services	DaaS: Cloud-based mobile Web Service Discovery	Utilização de Web Semântica para seleção de informações de Web Services no registro UDDI	eHealth Service Discovery Framework for a Low Infrastructure Context
Utiliza serviços web baseados em REST	**S**	S	NA		P
Utiliza serviços web baseados em SOAP	**S**		NA	S	
Utiliza web semântica	**S**	S		S	S
Utiliza ontologia própria				S	S
Utiliza ontologia OWL-S	**S**	NA	NA	NA	NA
Suporta descritor de serviços WSDL 1.0 ou 1.1	**S**	NA	P	S	NA
Suporta descritor de serviços WSDL 2.0	**S**	NA	P		NA
Suporta descritor de serviços WADL		NA	P		NA
Utiliza framework para gerenciar contexto	**S**			NA	
Suporta atributos de QoS	**P**	P		NA	
Suporta contexto do usuário		NA		NA	S
Suporta contexto de rede	**S**	NA		NA	
Suporta contexto do dispositivo	**S**	NA		NA	
Utiliza contexto dinamicamente	**S**			NA	P
Descoberta de serviços puramente dinamica	**S**	P	P		P
Descoberta de serviços puramente estática		P	P		P
Utiliza UDDI como registro padrão de serviços	**S**	NA	NA	S	S
Realiza busca por palavras-chave	**P**	S	S	S	P
Realiza busca por conteúdo	**S**	P			S

Legenda:
S = Apresenta a característica
P = Parcialmente
NA = Não se aplica

Tabela 5.1: Análise comparativa da proposta com os trabalhos correlatos.

5.3 Considerações Finais

Este capítulo apresentou os trabalhos associados ao desenvolvimento de aplicações sensíveis ao contexto que possuem correlação ao trabalho desenvolvido nesta pesquisa.

O trabalho proposto é baseado em diversas características dos trabalhos correlatos, apresentando, porém, suas próprias contribuições por meio da adequação e evolução dos trabalhos em que está baseado. Mais precisamente, as contribuições deste trabalho estão voltadas para o suporte ao desenvolvimento de aplicações que utilizam serviços web como principal tecnologia. Para isso, no trabalho são combinadas as características de sensibilidade ao contexto, ontologias e serviços web, para realização da descoberta de serviços sensível ao contexto.

Capítulo 6

CONCLUSÃO E TRABALHOS FUTUROS

A Computação Ubíqua tem demandado aos engenheiros de software uma série de requisitos adicionais ao desenvolvimento. Dentre estes requisitos tem-se a necessidade de descobrir dinamicamente serviços web, para que a aplicação possa adaptar-se dinamicamente de acordo com a necessidade do usuário final da aplicação. A definição de uma arquitetura adequada, portanto, torna-se essencial para fornecer aos desenvolvedores o suporte necessário para o cumprimento dos requisitos de adaptação demandados por uma aplicação ubíqua, considerando-se os diferentes contextos em que se executa a aplicação.

Neste sentido, o estudo e utilização de novas técnicas para o desenvolvimento de aplicações que suportem diferentes contextos foi um dos principais pontos explorados neste trabalho, além da utilização de serviços para interoperabilidade dos sistemas e a utilização de tecnologias distintas para implementação desses serviços. Uma modelo arquitetural foi proposto com base nas concepções de Computação Orientada a Serviços (SOC) e Arquitetura Orientada a Serviços SOA. O modelo proposto define meios de publicação e descoberta de serviços web, com base em ontologias e contexto computacional.

Com objetivo de testar a aplicabilidade e quantificar os resultados por meio da utilização da aplicação, testes automatizados foram construídos, assim como um experimento com desenvolvedores de software.

A principal contribuição do UDDI Semântico, para essa pesquisa e em outros contextos, é a busca de serviços por conteúdo, e também a reutilização de serviços, podendo publicá-los sem a necessidade de publicar a respectiva ontologia. Também a utilização de serviços baseados na arquitetura REST.

6.1 Contribuições

Uma contribuição importante foi desenvolvimento do módulo adicional para o framework Ubicon, que possibilita abstrair as tarefas de aquisição e manipulação do contexto, tanto do perfil do dispositivo, quanto do perfil da rede de acesso, permitindo que o cliente execute sempre o melhor serviço para seu contexto atual. Deste modo, o desenvolvimento torna-se simplificado, uma vez que o desenvolvedor não precisa se preocupar com essas tarefas. O objetivo do Ubi-Con Estendido é fornecer aos desenvolvedores um "esqueleto"que possa ser instanciado para o desenvolvimento de aplicações ubíquas através da combinação de diferentes perfis recuperados do contexto (e.g. dispositivo, usuário, rede).

Outra contribuição foi o desenvolvimento do UDDI Semântico, o qual visa armazenar serviços de diversas entidades, facilitando a descoberta e invocação dos mesmos, através de ontologias, geradas automaticamente por algoritmos desenvolvidos neste trabalho que utilizam algumas APIs auxiliares de manipulação de ontologias e descritores de serviços. Essa ferramenta oferece um interface gráfica, baseada em ambiente web e também Web Services para publicação e descoberta de serviços.

E, como principal contribuição deste trabalho, foi realizada a pesquisa e o desenvolvimento do algoritmo de descoberta, este, que por sua vez é dinâmico, sensível ao contexto e realiza busca por palavras-chave e conteúdo, com auxilio das ontologias geradas.

6.2 Limitações

Algumas limitações deste trabalho foram identificadas a partir de uma análise crítica do trabalho desenvolvido.

Na avaliação da arquitetura proposta foram utilizados serviços desenvolvidos propositalmente para o ambiente de testes. Apesar desses serviços funcionarem perfeitamente e criarem composições entre eles uma das limitações da avaliação deste trabalho é que ela foi baseada em poucos serviços, fazendo necessária a criação de um maior número de serviços para testar a capacidade real da arquitetura e de seus componentes.

Outra limitação deste trabalho refere-se às conclusões obtidas na avaliação apresentada no Capítulo 4. Apesar do estudo ter apontado ganhos de tempo em relação a descoberta automática do serviço adequado ao contexto do usuário, em conformidade com os benefícios que já eram esperados, deve-se considerar que o fenômeno observado limita-se ao escopo do ambiente de testes automatizados. Por questões de validade, para estender a amplitude dos resultados ob-

tidos para um contexto mais amplo, se faz necessário que novos estudos em ambiente *in-vivo*, comparando-se também com outras ferramentas e abordagens de publicação e descoberta de serviços.

Na avaliação do framework UbiCon Estendido, considerou-se o comportamento da descoberta de serviço, verificando se a mesma ocorre em conformidade com o perfil da rede de acesso identificado no contexto, bem como sua capacidade de carga no servidor, em termos de tempo de resposta, utilizando emuladores e usuários virtuais simultâneos. Uma outra avaliação mais detalhada, que não foi realizada neste trabalho, refere-se ao estudo do desempenho do framework utilizando um conjunto maior de serviços e dispositivos reais de acesso, em diferentes contextos de redes.

6.3 Trabalhos Futuros

Ao longo do desenvolvimento deste trabalho, algumas oportunidades de melhoria, tanto do suporte computacional quanto das avaliações, foram destacadas. Além disso, novas oportunidades de pesquisa foram identificadas. Os possíveis trabalhos futuros envolvem:

- Criação de testes automatizados, unitários e de comportamento, possibilitando encontrar possíveis erros.
- Melhorar a interface web criada, possibilitando a inclusão de mais provedores e mais serviços.
- Realizar estudo de caso com uma base maior de serviços, verificando se o algoritmo é estável com uma quantidade maior de serviços.
- Nova extensão do framework UbiCon de forma a contemplar outras características contextuais mais dinâmicas acerca do dispositivo e da rede, além da inclusão de outras fontes de contexto associadas a outras entidades, tais como o usuário, tempo.
- Criação de algoritmos de seleção e composição de serviços baseados na descoberta dos serviços que é sensível ao contexto.
- Incluir busca por similaridade das palavras-chave do serviço, utilizando o framework WordNet.

Por fim, observou-se que, apesar do foco deste trabalho ter sido mantido no domínio da publicação e descoberta de serviços a arquitetura proposta possibilita a extensão e reúso de

outros artefatos, podendo agregar novos algoritmos, e incrementar o repositório semântico, a fim de adaptar dinamicamente uma composição de serviços, que não satisfaçam o contexto do usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTONIOU, G.; Van Harmelen, F. *A semantic web primer*. [S.l.]: MIT press, 2004.

BALDAUF, M.; DUSTDAR, S.; ROSENBERG, F. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, Inderscience, v. 2, n. 4, p. 263–277, 2007.

BELLUR, U.; VADODARIA, H. Web Service Ranking Using Semantic Profile Information. In: *Web Services, 2009. ICWS 2009. IEEE International Conference on*. [S.l.: s.n.], 2009. p. 872–879.

BELQASMI, F.; GLITHO, R.; FU, C. RESTful web services for service provisioning in next-generation networks: A survey. *IEEE Communications Magazine*, v. 49, n. 12, p. 66–73, 2011.

BELQASMI, F. et al. SOAP-based vs. RESTful web services: A case study for multimedia conferencing. *IEEE Internet Computing*, v. 16, n. 4, p. 54–63, 2012.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. *Scientific american*, New York, NY, USA:, v. 284, n. 5, p. 28–37, 2001.

CARTLIDGE, A. et al. *An Introductory Overview of ITIL V3*. [S.l.]: ITSMF, 2009. (The IT Infrastructure Library). ISBN 9780955124587.

CASAL, J. A. et al. Formalising the software evaluation process. In: IEEE. *Computer Science, 1998. SCCC'98. XVIII International Conference of the Chilean Society of*. [S.l.], 1998. p. 15–24.

CHEN, H.; FININ, T.; JOSHI, A. An intelligent broker for context-aware systems. In: *Adjunct proceedings of Ubicomp*. [S.l.: s.n.], 2003. v. 3, p. 183–184.

CIRILO, C. E. et al. A hybrid approach for adapting web graphical user interfaces to multiple devices using information retrieved from context. In: *DMS 2010 - Proceedings of the 16th International Conference on Distributed Multimedia Systems*. [S.l.: s.n.], 2010. p. 168–173.

Curbera, E. C. F.; Weerawarana, G. M. S. *Web Services Description Language (WSDL)*. 2001. Disponível em: <http://www.w3.org/TR/wsdl>.

DEY, A. K. *Providing architectural support for building context-aware applications*. Tese (Doutorado) — Georgia Institute of Technology, 2000.

DEY, A. K. Understanding and using context. *Personal and ubiquitous computing*, Springer, v. 5, n. 1, p. 4–7, 2001.

Dos Santos, V. V. *CEManTIKA: A domain-independent framework for designing context-sensitive systems*. Tese (Doutorado) — Universidade Federal de Pernambuco, 2008.

DUSTDAR, S.; SCHREINER, W. A survey on web services composition. *International Journal of Web and Grid Services*, Inderscience, v. 1, n. 1, p. 1–30, 2005.

ELGAZZAR, K.; HASSANEIN, H. S.; MARTIN, P. Daas: Cloud-based mobile web service discovery. *Pervasive and Mobile Computing*, Elsevier, 2013.

ERL, T. *SOA: Princípios de Design de Serviços*. [S.l.]: Prentice Hall, 2009.

FARRAG, T. A.; SALEH, A. I.; ALI, H. A. Toward swss discovery: mapping from wsdl to owl-s based on ontology search and standardization engine. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 25, n. 5, p. 1135–1147, 2013.

FERREIRA FILHO, O. F.; VARELLA, M. A. G. *Serviços semânticos: uma abordagem RESTful*. Tese (Doutorado) — Dissertação de mestrado, USP, 2010. Disponível em< http://www.teses.usp.br/teses/disponiveis/3/3141/tde-11082010-120409/pt-br. php>. Acessado em 3 de Fevereiro de 2013, 2011.

FIELDING, R. T.; TAYLOR, R. N. Principled design of the modern web architecture. In: *Proceedings - International Conference on Software Engineering*. [S.l.: s.n.], 2000. p. 407–416.

FILHO, O. F. F.; FERREIRA; VARELLA, M. A. G. SEMANTIC WEB SERVICES: A RESTFUL APPROACH. *Proceedings of the IADIS International Conference on WWW/Internet*, p. 169–180, 2009.

FORTE, M.; SOUZA, W. L. de; PRADO, A. F. do. Using ontologies and Web services for content adaptation in Ubiquitous Computing. *Journal of Systems and Software*, v. 81, n. 3, p. 368–381, 2008.

GRöNROOS, C. *Marketing - Gerenciamento e Serviços*. [S.l.]: Editora Campus, 2009. ISBN 8535232060.

GRUBER, T. R. et al. Toward principles for the design of ontologies used for knowledge sharing. *International journal of human computer studies*, London; San Diego: Academic Press, c1994-, v. 43, n. 5, p. 907–928, 1995.

Haas, H.; Allen Brown. *Web Services Glossary*. 2004. Disponível em: <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>.

HADLEY, M. *Web Application Description Language*. 2009. Disponível em: <http://www.w3.org/Submission/wadl>.

HENDLER, J. Agents and the semantic web. *Intelligent Systems, IEEE*, IEEE, v. 16, n. 2, p. 30–37, 2001.

HERMAN, I. *OWL: Web Ontology Language Reference*. 2004. Disponível em: <http://www.w3.org/2004/OWL/>.

ISOTANI, S. et al. Estado da arte em web semântica e web 2.0: potencialidades e tendências da nova geração de ambientes de ensino na internet. *Revista Brasileira de informática na educação*, v. 17, n. 01, p. 30, 2009.

JIANG, W.; LEE, D.; HU, S. Large-scale longitudinal analysis of SOAP-based and RESTful web services. In: *Proceedings - 2012 IEEE 19th International Conference on Web Services, ICWS 2012*. [S.l.: s.n.], 2012. p. 218–225.

JOHN, D.; RAJASREE, M. A framework for the description, discovery and composition of restful semantic web services. In: ACM. *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*. [S.l.], 2012. p. 88–93.

LUSCH, R. F.; VARGO, S. *The Service-dominant Logic of Marketing: Dialog, Debate, And Directions*. Sharpe, 2006. ISBN 9780765614902. Disponível em: <http://books.google.com.br/books?id=Sdn3ZK5PUoEC>.

MANOLA, F.; MILLER, E.; MCBRIDE, B. Rdf primer. *W3C recommendation*, v. 10, p. 1–107, 2004.

MARTIN, D. et al. *OWL-S: Semantic Markup for Web Services*. 2004. Disponível em: <http://www.w3.org/Submission/OWL-S/>.

MCGUINNESS, D. L.; HARMELEN, F. V. et al. Owl web ontology language overview. *W3C recommendation*, v. 10, n. 2004-03, p. 10, 2004.

MCILRAITH, S. A.; MARTIN, D. L. Bringing semantics to web services. *Intelligent Systems, IEEE*, IEEE, v. 18, n. 1, p. 90–93, 2003.

Mitra, N.; Lafon, Y. *SOAP Version 1.2 Part 0: Primer (Second Edition)*. 2007. Disponível em: <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>.

MONTGOMERY, D. C. *Design and analysis of experiments*. [S.l.]: John Wiley & Sons, 2008.

Moreau, R. C. J.-J.; Weerawarana, A. R. S. *Web Services Description Language (WSDL) Version 2.0*. 2007. Disponível em: <http://www.w3.org/TR/wsdl20>.

NAKAMURA, L. H. V. *Utilização de web semântica para seleção de informações de web services no registro UDDI uma abordagem com qualidade de serviço*. Tese (Doutorado) — Universidade de São Paulo, 2012.

OASIS. *OASIS UDDI Specificaion TC*. 2002. Disponível em: <https://www.oasis-open.org/committees/uddi-spec/faq.php>.

PAPAZOGLOU, M. P. Service-oriented computing: Concepts, characteristics and directions. In: IEEE. *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*. [S.l.], 2003. p. 3–12.

PAPAZOGLOU, M. P.; HEUVEL, W.-J. V. D. Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*, Springer, v. 16, n. 3, p. 389–415, 2007.

PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. RESTful web services vs. "Big"web services: Making the right architectural decision. In: *Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08*. [S.l.: s.n.], 2008. p. 805–814.

PERERA, C. et al. Context aware computing for the internet of things: A survey. IEEE, 2013.

PRESSMAN, R. S. *Engenharia de software*. 6. ed. São Paulo: McGraw-Hill, 2006.

RAMASWANY, R. *Design and management of service processes: keeping customers for life*. [S.l.]: Addison-Wesley, 1996. ISBN 8535232060.

REDDY, P. R.; DAMODARAM, A. Web services discovery based on semantic similarity clustering. In: *2012 CSI 6th International Conference on Software Engineering, CONSEG 2012*. [S.l.: s.n.], 2012.

RODRIGUEZ, A. *RESTful Web services: The basics*. 2008. Disponível em: <http://www.ibm.com/developerworks/webservices/library/ws-restful>.

ROSAS, E. et al. Web search results caching service for structured p2p networks. *Future Generation Computer Systems*, Elsevier, v. 30, p. 254–264, 2014.

SANTOS, P. B.; WIVES, L. K.; DE OLIVEIRA, J. P. M. An improved approach for measuring similarity among semantic Web services. In: *WEBIST 2012 - Proceedings of the 8th International Conference on Web Information Systems and Technologies*. [S.l.: s.n.], 2012. p. 83–88.

SAÚDE, O. M. da. *A Glossary of Terms for Community Health Care and Services for Older Persons*. [S.l.]: WHO, 2004. (Ageing and health technical report).

SHENG, Q. Z. et al. Web services composition: A decade?s overview. *Information Sciences*, Elsevier, v. 280, p. 218–238, 2014.

SILVA, E. A. N. d.; LUCRÉDIO, D. Software engineering for the cloud: a research roadmap. In: IEEE. *Software Engineering (SBES), 2012 26th Brazilian Symposium on*. [S.l.], 2012. p. 71–80.

SILVA, E. G. da; PIRES, L. F.; SINDEREN, M. van. Towards runtime discovery, selection and composition of semantic services. *Computer communications*, Elsevier, v. 34, n. 2, p. 159–168, 2011.

SILVA, V. Gomes da et al. An approach to dynamic discovery of context-sensitive web services. In: *ICIW 2013, The Eighth International Conference on Internet and Web Applications and Services*. [S.l.: s.n.], 2013. p. 83–89.

TEAM, C. P. Cmmi for development, version 1.2. 2006.

TEGEGNE, T.; KANAGWA, B.; WEIDE, T. van der. ehealth service discovery framework for a low infrastructure context. In: IEEE. *Computer Technology and Development (ICCTD), 2010 2nd International Conference on*. [S.l.], 2010. p. 606–610.

TEGEGNE, T.; KANAGWA, B.; WEIDE, T. van der. Service discovery framework for personalized ehealth. In: *Int. Conf. on Services Science, Management and Engineering, SSME*. [S.l.: s.n.], 2010. p. 26–28.

TEGEGNE, T.; WEIDE, T. van der. ehealth service discovery framework: a case study in ethiopia. In: *Information Quality in e-Health*. [S.l.]: Springer, 2011. p. 397–416.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. *Introdução à engenharia de software experimental*. [S.l.]: UFRJ, 2002.

VIEIRA, R. et al. W eb semântica: ontologias, lógica de descrição e inferência. *Web e Multimidia: desafios e soluções. Porto Alegre: SBC*, p. 127–167, 2005.

VIEIRA, V.; CALDAS, L. R.; SALGADO, A. C. Towards an Ubiquitous and Context sensitive public transportation system. In: *Proceedings - 4th International Conference on Ubi-Media Computing, U-Media 2011*. [S.l.: s.n.], 2011. p. 174–179.

VIEIRA, V.; TEDESCO, P.; SALGADO, A. C. Designing context-sensitive systems: An integrated approach. *Expert Systems with Applications*, v. 38, n. 2, p. 1119–1138, 2011.

VINOSKI, S. Serendipitous reuse. *IEEE Internet Computing*, v. 12, n. 1, p. 84–87, 2008.

WEERAWARANA, S. et al. *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. [S.l.]: Prentice Hall PTR, 2005.

WEISER, M. The computer for the 21 st century. *ACM SIGMOBILE mobile computing and communications review*, ACM, v. 3, n. 3, p. 3–11, 1999.

WOHLIN, C. et al. *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000. ISBN 0-7923-8682-5.

WOHLIN, C. et al. *Experimentation in software engineering*. [S.l.]: Springer, 2012.

GLOSSÁRIO

ASC – *Aplicação Sensível ao Contexto*

CEManTIKA – *Contextual Elements Modeling and Management through Incremental Knowledge Acquisition*

DaaS – *Discovery as a Service*

EC – *Element Contextual*

EC – *Elemento contextual*

EE – *Entreprise Edition*

EICAF – *Extend Internet Content Adaptation Framework*

HEW – *Health Extension Workers*

HSS – *Health Service System*

HTTP – *Hypertext Transfer Protocol*

IOPE – *Input Output Precondition Effect*

JUDDI – *Java implementation of the Universal Description, Discovery, and Integration - Judy*

LOC – *Lines Of Code*

MDD – *Model-Driven Development*

OASIS – *Organization for the Advancement of Structured Information Standards*

OWL-S – *Ontology Web Language for Web Services*

OWL – *Ontology Web Language*

QoS – *Quality of Service*

QoS – *Quality of service*

RDF – *Resource Description Framework*

REST – *REpresentational State Transfer*

RPC – *Remote procedure call*

SE – *Sistemas Embarcados*

SOAP – *Simple Object Access Protocol*

SOA – *Service-Oriented Architecture*

SOC – *Service-Oriented Computing*

SSC – *Sistemas Sensíveis ao Contexto*

SWRL – *Semantic Web Rule Language*

SaaS – *Software as a Service*

TI – *Tecnologia da Informação*

UDDI – *Universal Description Discovery and Integration*

UDOnt-Q – *Universal Discovery with Ontology and QoS*

UFSCar – *Universidade Federal de São Carlos*

URI – *Uniform Resource Identifier*

UbiCon – *Ubiquitous Context Framework*

W3C – *World Wide Web Consortium*

WADL – *Web Application Description Language*

WSDL – *Web Service Description Language*

WS – *Web Service*

XML – *eXtensible Markup Language*

Apêndice A

FORMULÁRIO DE CARACTERIZAÇÃO DO PARTICIPANTE

Este formulário tem por objetivo caracterizar sua experiência acadêmica, pessoal e profissional com relação à Ciência da Computação. Por favor, responda a TODAS as questões o mais fielmente possível. Toda informação fornecida é confidencial, sendo que seu nome e quaisquer outros meios de identificação não serão divulgados em nenhuma hipótese.

1. Dados do participante

Nome: _____

Idade: _____

2. Formação acadêmica

( ) Graduação
( ) Mestrado
( ) Doutorado
Ano de Ingresso: _____
Mês/Ano de Conclusão (ou previsão de conclusão): ____/_____

3. Experiência Profissional relacionadas a conceitos e técnicas empregados no experimento

Esta seção será utilizada para compreender sua familiaridade com os conceitos e técnicas que serão utilizados nas atividades do experimento.

3.1 Assinale a opção que melhor refilta seu grau de experiência com as tecnologias listadas a seguir, considerando a escala de 5 pontos:

0 = Nenhum. Eu não tenho familiaridade com este assunto. Eu nunca vi/fiz isto.
1 = Já li ou estudei sobre isto. Conheço os conceitos/técnicas.
2 = Eu utilizo algumas vezes em projetos pessoais, mas não sou especialista.
3 = Eu sou muito familiar com este assunto. Eu me sentiria confortável fazendo isto.
4 = Eu utilizo em grande parte dos projetos que realizo.

Linguagem de Programação Java	0	1	2	3	4
eXtensible Markup Language (XML)	0	1	2	3	4
Web Services Description Language (WSDL)	0	1	2	3	4
Web Ontology Language (OWL)	0	1	2	3	4
Semantic Markup Language for Web Services (OWL-S)	0	1	2	3	4

Banco de Dados MySql	0	1	2	3	4
IDE Eclipse	0	1	2	3	4
Aplicações Sensíveis ao Contexto	0	1	2	3	4
Ontologias	0	1	2	3	4
Redes e QoS	0	1	2	3	4

3.2 Qual das opções a seguir melhor descreve sua experência anterior com relação ao desenvolvimento de software na prática?

Tenho desenvolvido software por conta própria (sozinho) ( )	Tenho desenvolvido software como membro de equipe durante cursos que realizo ( )	Tenho desenvolvido software como membro de equipe, na indústria, há menos de 2 anos ( )	Tenho desenvolvido software como membro de equipe, na indústria, há 2 anos ou mais ( )

3.2.1 Como você distribuiria o tempo gasto em sua experiência com programação?

_____ % em esforço individual (sozinho)
_____ % em desenvolvimento conjunto com outras pessoas (equipe)
_____ % na supervisão de outros programadores (gerência)
100 % ← TOTAL

3.3 Qual sua habilidade em …

a) desenvolver software para Web?
( ) Especialista ( ) Avançado ( ) Médio ( ) Básico ( ) Nenhum
Meses de experiência: ____ (como aluno em um curso) ____ (na indústria)

b) desenvolver software com Web Semântica?
( ) Especialista ( ) Avançado ( ) Médio ( ) Básico ( ) Nenhum
Meses de experiência: ____ (como aluno em um curso) ____ (na indústria)

c) desenvolver Web Services?
( ) Especialista ( ) Avançado ( ) Médio ( ) Básico ( ) Nenhum
Meses de experiência: ____ (como aluno em um curso) ____ (na indústria)

d) aplicações sensíveis ao contexto?
( ) Especialista ( ) Avançado ( ) Médio ( ) Básico ( ) Nenhum
Meses de experiência: ____ (como aluno em um curso) ____ (na indústria)

Obrigado por sua colaboração!

Apêndice B

FORMULÁRIO DE CONSENTIMENTO

Experimento

Este experimento visa avaliar a aplicação desenvolvida para apoiar a descoberta e publicação de serviços sensível ao contexto.

Idade

Eu declaro ser maior de 18 (dezoito) anos de idade e concordar em participar do experimento conduzido por Victor Gomes da Silva na Universidade Federal de São Carlos (UFSCar).

Procedimento

Este experimento ocorrerá em uma única sessão, que incluirá o desenvolvimento de abstrações/descrições de Web Services (WSs), que formam uma composição entre eles. O desenvolvimento dos WSs se dará com auxílio de um editor de textos e após serão submetidos a aplicação proposta, no caso UDDI Semântico. Eu entendo que, uma vez que o experimento tenho sido concluído, os trabalhos que desenvolvi, bem como os dados coletados, serão estudados visando analisar a aplicação proposta.

Confidencialidade

Estou ciente de que toda informação coletada neste experimento é confidencial, e meu nome ou quaisquer outros meios de identificação não serão divulgado. Da mesma forma, me comprometo a não comunicar meus resultados aos demais participantes e/ou a outros grupos enquanto não terminar o experimento, bem como manter sigilo das técnicas e documentos apresentados que fazem parte do experimento.

Benefícios e liberdade de desistência

Eu entendo que os beneficíos que receberei deste experimento se limitam ao aprendizado do material que é distribuído e apresentado. Eu compreendo que sou livre para realizar perguntas a qualquer momento, solicitar que qualquer informação relacionada à minha pessoas não seja incluída no experimento, ou comunicar minha desistência de participação. Eu entendo que participo livremente com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

Pesquisador Responsável
Victor Gomes da Silva
Programa de Pós-Graduação em Ciência da Computação - PPG-CC/DC/UFSCar

Professor Responsável
Prof. Dr. Antonio Francisco do Prado
Programa de Pós-Graduação em Ciência da Computação - PPG-CC/DC/UFSCar

Nome (em letra de forma):

Assinatura:

Data: ____/____/_____

Apêndice C

DESCRIÇÃO DA TAREFA

Instruções

Este experimento avaliará a aplicação proposta que server como um repositório semântico de web services. A análise dos resultados será baseada nos dados coletados durante a execução das atividades propostas no experimento. Assim, de forma que o experimentador possa melhor avaliar os resultados obtidos, nos formulários entregues a você preencha corretamente todos os dados relacionados ao horário de início e término de cada atividade, bem como os dados relacionados ao número de linhas de código implementadas.

Da mesma maneira, caso encontre algum problema de ordem técnica durante a execução de determinada atividade, anote os horários de identificação e solução do problema juntamente com sua descrição no formulário apropriado.

Pergunte e comente tudo o que julgar necessário.

Contextualização

Atualmente a Internet é um dos maiores meios de comunicação do mundo, transformando e acelerando as formas de publicação e consumo de conteúdo. Ao mesmo tempo, usuários têm se tornado mais exigentes pelo conteúdo e pelo consumo do grande volume de informação que a Web proporciona. Desde a sua criação, a Internet passou por algumas fases, mostrando que as mudanças acontecem de acordo com as exigências dos usuários, como por exemplo, a evolução da Web para Web 2.0 que é marcada pela interação, colaboração e comunicação entre os usuários, estes podendo publicar e compartilhar seus próprios conteúdos na Internet. Esta realidade tem levado a necessidade de construir aplicações que fornecem suporte a grande quantidade de informações para usuários e dispositivos distintos, dispostos em múltiplos contextos de uso, aumentando a demanda por software na computação ubíqua, onde o acesso às aplicações ocorre de qualquer lugar, a qualquer momento e a partir de diferentes dispositivos. A construção e manutenção de versões específicas das aplicações para cada contexto de uso tornaram-se difíceis devido à diversidade de dispositivos, usuários, redes de acesso e outros fatores a serem considerados. Além disso, é necessário que as aplicações sejam desenvolvidas de modo que a compatibilidade às características contextuais do ambiente em que operam seja mantida, não prejudicando a interação dos usuários. Considerando este problema uma solução muito utilizada é o uso da Arquitetura Orientada a Serviços, juntamente com tecnologias de Web Services, que tem a intenção de integrar sistemas heterogêneos e disponibilizar aplicações na forma de serviços com o uso de padrões abertos de Internet, favorecendo o reuso e manutenção de software. Um desafio para Engenharia de Software, portanto, é prover métodos e ferramentas adequadas para construção de aplicações ubíquas adaptáveis a diferentes contextos, atendendo à diversidade de dispositivos, preferências de usuários, arquiteturas de redes de acesso e outros contextos. Este desafio tem despertado o interesse de diversos pesquisadores ao longo dos últimos anos, que propõem soluções que orientam o desenvolvimento de aplicações

sensíveis ao contexto. Motivados em superar esse desafio, com o uso de Web Services, esta dissertação apresenta uma solução que viabiliza a descoberta de Web Services, em tempo de execução, de acordo com mudanças de contexto.

Tarefa 1 - Desenvolver, Publicar e Descobrir os Serviços

Você recebeu a descrição de um exemplo de aplicação ubíqua sensível ao contexto que utiliza serviços web como principal tecnologia. Uma vez que o foco deste experimento é avaliar os serviços de publicação e descoberta de Web Services em relação a outra abordagem (jUDDI), sua tarefa é **desenvolver 3 Web Services que juntos formam um serviço maior, publicá-los utilizando a interface web do UDDI Semântico e descobri-los utilizando a ferramenta de testes jMeter.**

Utilize o material de apoio para auxilia-lo na execução das atividades. Tendo que os requisitos da aplicação ainda não foram identificados, inicie a avaliação definindo-os. Para cada atividade realizada anote os dados temporais e de linhas de código no formulário apropriado.

Tarefa 2 - Testar Usabilidade do UDDI Semântico em relação ao jUDDI

Uma vez que o foco desta tarefa é avaliar a interface de publicação em relação a outra abordagem (jUDDI), sua tarefa é **utilizar a interface web tanto do UDDI Semântico quanto do jUDDI e informar se são similares, usuais e interativas.**

Utilize o material de apoio para auxilia-lo na execução desta atividade. Para cada atividade realizada anote os dados temporais e de linhas de código no formulário apropriado.

Ao final do experimento, compacte o projeto da aplicação desenvolvida e envie por email para v.gomes90@gmail.com, utilizando como assunto o seguinte padrão: "Experimento UDDI".

Apêndice D

DESCRIÇÃO DA APLICAÇÃO E MATERIAL DE APOIO

O Health Service System (HSS) consiste em um protótipo de um sistema Web de uso gratuito que tem o objetivo de facilitar e agilizar a marcação de consultas médicas pela Internet, levando em consideração a proximidade dos pacientes em relação as clínicas e centros de saúde de interesse. Desta forma, médicos e centros de saúde especializados que desejam oferecer marcação online de serviços de saúde a pacientes de sua região e pessoas físicas que procuram por esses serviços podem fazê-lo por meio do HSS.

Breve descrição do funcionamento do sistema

O Paciente cadastra seu perfil com suas preferências; a partir disso, quando o Paciente procurar um médico para marcar uma consulta, o sistema age sozinho; o Paciente pode informar sua enfermidade ou a especialidade médica requerida para filtrar a busca por clínicas e centros apropriados. O sistema procura/localiza clinicas próximas ao paciente, procura/localiza médicos com as devidas especialidades, a partir das preferências do paciente, verifica a agenda do médico e marca a consulta. O sistema informa ao médico e paciente, através de mensagens, o resultado do agendamento da consulta.

Material de Apoio

Alguns casos de uso identificados para o HSS são apresentados no diagrama que está em anexo. Para atender aos propósitos deste experimento, foque no desenvolvimento dos serviços relacionados à algum desses casos de uso.

Exemplo de Web Service
Os exemplos de código-fonte wsdl encontram-se no endereço:
http://www2.dc.ufscar.br/~victor.silva/experimento/wsdl

Realizando a publicação do serviço no UDDI
Após o desenvolvimento dos serviços, será realizada a publicação do mesmo. Para realizar a publicação acesse este endereço: **http://semanticuddi-vgomes.rhcloud.com/** **.**

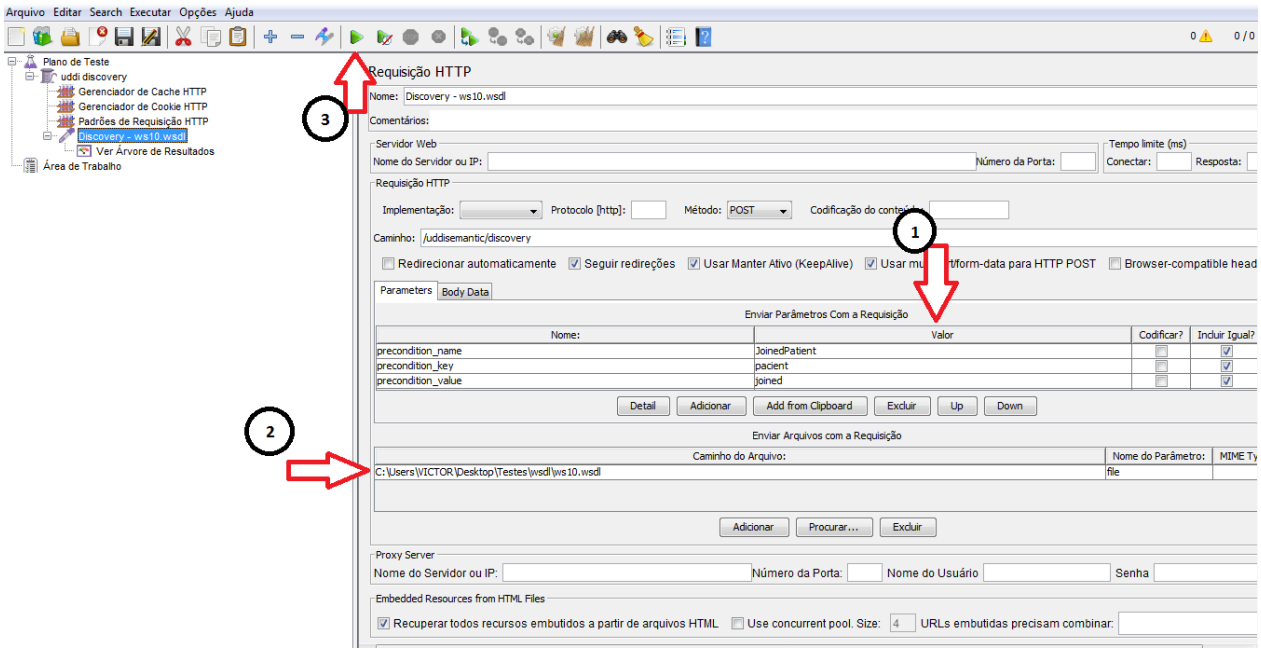
Clique no selecione o WSDL e envie-o. Logo após, informe o contexto do serviço e salve-o na base de dados.

Realizando a descoberta

Para realizar a descoberta dos serviços, abra a ferramenta jMeter e importe o arquivo que se encontra no endereço - http://www2.dc.ufscar.br/~victor.silva/experimento/descoberta.

1. Na ferramenta de teste jMeter vá no menu *Arquivo* e escolha a opção *Abrir*
2. Na janela que se abre, escolha o arquivo uddi_discovery.jmx
3. Pressione o botão *Abrir.*
4. Adicione os valores aos parametros de precondição e efeitos.
5. Adicione o arquivo WSDL que representa o serviço fora do contexto.
6. Finalmente clique no botão *Iniciar*, que se encontra no meu superior, representado por uma seta verde.

E configurem os parametros da requisição do serviço, juntamente com o wsdl que se encontra fora do contexto:



Analisando a interface web do UDDI Semântico
http://semanticuddi-vgomes.rhcloud.com/ .

Analisando a interface web do jUDDI
http://uddi-jbossoverlord.rhcloud.com/ .

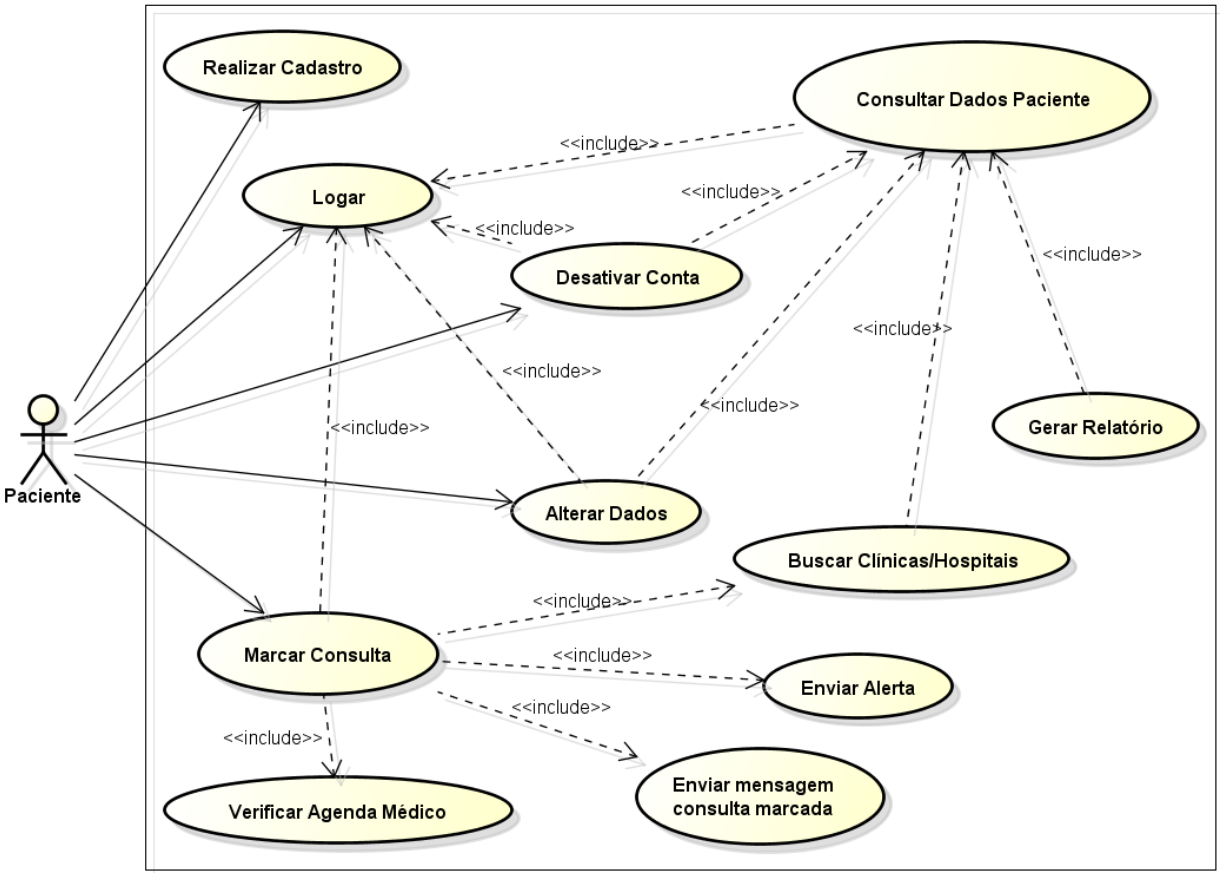


Diagrama de casos de uso do HSS

Apêndice E

FORMULÁRIO DE COLETA DE DADOS

Nome:

Anote na tabela abaixo os horários de início e fim de cada atividade realizada.

Atividade	Hora Início	Hora Fim
Projetar	___:___ h	___:___ h
Implementar	___:___ h	___:___ h
Testar (Publicação e Descoberta)	___:___ h	___:___ h

Anote as quantidades de linha de código geradas manualmente.

Linhas de código geradas **manualmente**: _____

Anote os problemas técnicos encontrados durante a execução do experimento, indicando o horário de identificação e resolução do problema, bem como sua breve descrição.

Descrição do Problema	Hora de Identificação	Hora de Resolução
	___:___ h	___:___ h
	___:___ h	___:___ h
	___:___ h	___:___ h
	___:___ h	___:___ h
	___:___ h	___:___ h

Apêndice F

FORMULÁRIO DE AVALIAÇÃO

Nome:

1. Você considera que a aplicação auxiliou na publicação de serviços web? Justifique.
( ) Sim ( ) Não ( ) Parcialmente

2. Você sentiu dificuldades na realização das atividades? Especifique.
( ) Sim ( ) Não ( ) Parcialmente

3. Quais sugestões você teria para melhorar na aplicação?

4. Você acredita que o uso de um repositório semântico facilitou a descoberta de serviços? Justifique.
( ) Sim ( ) Não ( ) Parcialmente

5. Você considera viável, sob o ponto de vista de esforço de desenvolvimento e eficiência na descoberta de serviços, adotar a estratégia de publicação num repositório semântico? Justifique.
( ) Sim ( ) Não ( ) Parcialmente

6. Sobre a utilização da aplicação jUDDI**.** Qual interface você considera mais usual, do ponto de vista da publicação e descoberta de serviços? Justifique.
( ) UDDI Semântico ( ) jUDDI

7. Ainda sobre a utilização da aplicação jUDDI**.** Qual sistema lhe da mais funcionalidades? Considere o ponto de vista da publicação e descoberta de serviços? Justifique.
( ) UDDI Semântico ( ) jUDDI

Obrigado por sua colaboração!