Universidade Federal de São Carlos
Centro de Ciências Exatas e Tecnologia
Departamento de Ciência da Computação
Programa de Pós-Graduação em Ciência da Computação

"ADeSCoU: Uma Abordagem para o Desenvolvimento de Software para Computação Ubíqua"

Luiz Henrique Zambom Santana

São Carlos – SP

Maio/2008

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

S232ab

Santana, Luiz Henrique Zambom.
ADESCOU : Uma Abordagem para o desenvolvimento de software para computação ubíqua / Luiz Henrique Zambom Santana. -- São Carlos : UFSCar, 2008.
105 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2008.

1. Software - desenvolvimento. 2. Computação ubíqua. I. Título.

CDD: 004.21 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"AdeSCoU: Uma Abordagem para o Desenvolvimento de Software na Computação Ubíqua"

LUIZ HENRIQUE ZAMBOM SANTANA

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

Membros da Banca:

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

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

Prof. Dr. José Carlos Maldonado
(ICMC/USP)

Prof. Dr. Ivan Torres Pisa
(UNIFESP)

São Carlos
Maio/2008

Sonhar
Mais um sonho impossível
Lutar
Quando é fácil ceder
Vencer
O inimigo invencível
Negar
Quando a regra é vender
Sofrer
A tortura implacável
Romper
A incabível prisão
Voar
Num limite improvável
Tocar
O inacessível chão
É minha lei, é minha questão
Virar esse mundo
Cravar esse chão
Não me importa saber
Se é terrível demais
Quantas guerras terei que vencer
Por um pouco de paz
E amanhã, se esse chão que eu beijei
For meu leito e perdão
Vou saber que valeu delirar
E morrer de paixão
E assim, seja lá como for
Vai ter fim a infinita aflição
E o mundo vai ver uma flor
Brotar do impossível chão

Chico Buarque e Ruy Guerra

Agradecimentos

São muitas pessoas a quem devo agradecimentos. Primeiramente, aos meus pais, começo e fim de tudo na minha vida. Aos meus irmãos à quem espero contribuir, talvez como um contra exemplo, para que sejam melhores e mais felizes. Aos amigos antigos, principalmente os de Lins, e aos amigos feitos durante o mestrado, em particular: Diogo Martins, Caroline Perlin, Raphael Pereira, Rafael Milani e David Lorente. Agradeço também ao Prof. Carlos Roberto Valêncio, meu primeiro orientador, pela confiança depositada ao me indicar para o mestrado da UFSCar. Entretanto, com certeza à quem devo mais agradecimentos é aos professores Antonio Francisco do Prado, Wanderley Lopes de Souza e Mauro Biajiz. Com eles, aprendi muito mais do que Engenharia de Software, Redes de Computadores, Banco de Dados ou Computação Ubíqua. Aprendi a ler, organizar, criar, escrever e apresentar minhas idéias, em suma aprendi o que é pesquisa.

Sumário

Lista de Figuras

Lista de Tabelas e Listagens

Lista de Abreviações

AOSD	*Apect-Oriented Software Development*
ABP	*Aprendizado Baseado em Problemas*
B2B	*Business-to-Business*
CLDC	*Connected Limited Device Configuration*
ES	*Engenharia de Software*
FIPA	*Foundation for Intelligents Physical Agents*
GCU	*Grupo de Computação Ubíqua*
GPS	*Global Position System*
HTML	*HyperText Markup Language*
IHC	*Interação Humano-Computador*
IOC	*Initial Operational Capability*
JADE	*Java Agents Developement FramEwork*
JavaME	*Java MicroEdition*
LCA	*Life Clycle Architeture*
LCO	*Life Cycle Objective*
MAS-ML	*Multi-Agents Systems Modeling Language*
MDD	*Model-Driven Development*
MIDP	*Mobile Information Device Profile*
OMG	*Object Management Group*
OWL	*Ontology Web Language*
OWL-S	*OWL for Services*
PC	*Personal Computers*
PDA	*Personal Digital Assistants*

PDF	*Portable Document Format*
PDS	*Processo de Desenvolvimento de Software*
POCAp	*Process for Ontological Context-aware Applications*
PR	*Portfólio Reflexivo*
PRE	*Portfólio Reflexivo Eletrônico*
RMS	*Record Management System*
RUP	*Rational Unified Process*
SMA	*Sistemas Multi-Agentes*
UBICK	*Ubiquitous Computing Framework*
UbiComp	*Ubiquitous Computing*
UFSCar	*Universidade Federal de São Carlos*
UML	*Unified Modeling Language*
USF	*Unidade de Saúde da Família*
W3C	*Word Wide Web Consortium*
WML	*Wireless Markup Language*
WSDL	*Web Services Description Language*
WWW	*World Wide Web*

Resumo

Esta dissertação apresenta uma abordagem para orientar o desenvolvimento de software na Computação Ubíqua que instancia o *Rational Unified Process (RUP)*, com base em Ontologias, Serviços Web Semânticos, Agentes de Software. Ontologias são empregadas para facilitar o desenvolvimento de aplicações sensíveis a contexto. Serviços Web Semânticos são utilizados como componentes de software distribuídos pela Internet e são compostos para realizar tarefas complexas. Agentes de Software gerenciam as aplicações, facilitando o uso das Ontologias e dos Serviços Web Semânticos. Um framework, denominado UBICK, foi construído a fim de estruturar o projeto e implementação das aplicações ubíquas, baseado no reuso de componentes. Como estudo de caso, apresenta-se o uso da abordagem proposta no desenvolvimento de um Portfólio Reflexivo Eletrônico para o domínio de ensino na medicina.

Palavras-Chave: Computação Ubíqua, Ontologias, Serviços Web Semânticos e Agentes de Software.

Citação: Santana, Luiz H. Z. Uma abordagem para desenvolvimento de software na Computação Ubíqua. São Carlos, 2008. 97p. Dissertação de Mestrado - Departamento de Computação, Universidade Federal de São Carlos.

Abstract

This dissertation presents an approach to the software development in the Ubiquitous Computing, which instantiates the *Rational Unified Process (RUP)*, based on Ontologies, Semantic Web Services, and Software Agents. Ontologies are employed to improve the description of the use context, enabling the applications to be context-aware. Semantic Web Services are used as software components distributed over the Internet, and are composed to perform complex tasks. Software Agents manages the applications using the Ontologies and the Semantic Web Services. A framework, called UBICK, was built to facilitate the design and the implementation of ubiquitous applications, based on components reuse. A case study illustrates the use of this approach in order to develop an Electronic Reflexive Portfolio in the medical education domain.

Keywords: Ubiquitous Computing, Ontologies, Semantic Web Services and Software Agents.

Capítulo 1

Introdução

Pode-se classificar a história da Computação, em função dos ambientes computacionais predominantes, em três eras [HAN03]: a passada dos mainframes, a atual dos computadores pessoais *(PCs – Personal Computers)*, e a presente da Computação Ubíqua (UbiComp) ou Computação Pervasiva.

Para Mark Weiser, o pai da Computação Ubíqua, esta nova era caracteriza-se pela integração entre os computadores e o cotidiano das pessoas. Na visão de Weiser, a interação com computadores deveria ser tão relaxante quanto uma caminhada, já que existe mais informação em impressões digitais, durante esta caminhada, do que em qualquer sistema computacional. Dessa forma, integrar computadores ao ambiente humano, ao invés de forçar os humanos a integrarem-se ao ambiente computacional, tornaria seu o uso uma atividade menos frustrante [WEI91, WEI94].

Para atingir essa visão, as aplicações ubíquas devem radicalizar a descentralização do poder computacional, distribuindo-o através de uma grande variedade de dispositivos, que possuem funcionalidades específicas e assumem tarefas específicas. Cada um desses dispositivos contribui para a formação de um ambiente computacional globalmente heterogêneo, sendo que estes devem cooperar mutuamente, via padrões de comunicação, linguagens

de marcação e plataformas de software, visando à interoperabilidade entre os mesmos.

1.1. Caracterização do Problema

Vários avanços na pesquisa em Ciência da Computação contribuíram para maturidade tecnológica necessária para o desenvolvimento da visão de Weiser, dentre os quais destacam-se [SAH03]: o Computador Pessoal, que aproximou os computadores das pessoas, ainda que não tenha atingido todo o potencial da tecnologia da informação, causando um crescimento exponencial no desenvolvimento de componentes de hardware e de interfaces gráficas; a Computação Distribuída, que tornou os computadores conectados, possibilitando que compartilhassem suas capacidades através da rede, e permitiu aos usuários o acesso à informações e recursos remotos; e a Computação Móvel, que está tornando possível o uso de computadores “em qualquer lugar, a qualquer hora”, essencial para que este uso esteja disponível “em todas as horas, em todos os lugares”.

Entretanto, mesmo com a disponibilidade de tecnologias de hardware e redes, a visão de Weiser ainda não foi completamente atingida e o desenvolvimento de aplicações ubíquas ainda é manual e sob-demanda [HEL05]. Isto porque, a nova era da Computação Ubíqua apresenta desafios particulares para a Engenharia de Software, entre o quais destacam-se [ABO99, SAT01, KIN02, NIE04, END05, WAN05]: interoperabilidade, ciência de contexto, adaptação de conteúdo, capacidade de adaptação e segurança. Assim, a motivação desse trabalho é o desenvolvimento de uma abordagem para orientar o desenvolvimento de aplicações ubíquas considerando os desafios dessa nova era.

1.2. Proposta

Este trabalho propõe uma abordagem, denominada *Abordagem para o Desenvolvimento de Software na Computação Ubíqua (AdeSCoU)*, baseada no *Rational Unified Process (RUP)* [KRU03], que é combinado com Ontologias, Serviços Web Semânticos e Agentes de Software.

Ontologias são utilizadas para facilitar a descrição do contexto das aplicações da Computação Ubíqua. O contexto consiste de um conjunto de atributos que caracteriza as capacidades dos dispositivos de acesso, os dados pessoais e as preferências dos usuários, as condições da rede de acesso e as características dos conteúdos requisitados. Essas classes de informações são definidas em perfis dos dispositivos, dos usuários, das redes e dos conteúdos.

Serviços Web Semânticos são utilizados para implementar os componentes das aplicações desenvolvidos segundo a abordagem proposta, distribuindo sua execução e facilitando a composição de componentes complexos.

Agentes de Software são utilizados para gerenciar as aplicações, uma vez que podem oferecer benefícios como: comunicação sob-demanda, adaptação e configuração autônoma, aquisição de contexto, mobilidade e uso inteligente de Serviços Web Semânticos [SOL07].

Para apoiar o projeto e implementação das aplicações ubíquas foi construído um *framework*, denominado *UBIquitous Computing* frameworK *(UBICK)*, que estrutura as aplicações e oferece componentes previamente testados para realizar a comunicação entre clientes e servidores, ciência de contexto, *cache*, e outra tarefas. Assim, na abordagem AdeSCoU, o UBICK,

além de estruturar o desenvolvimento das aplicações, possibilita o reuso dos seus componentes, nas disciplinas de projeto e implementação.

1.3. Contribuições

A principal contribuição desse trabalho é a abordagem para o desenvolvimento de software na Computação Ubíqua, que diferencia dos demais processos de desenvolvimento pelo emprego combinado de Ontologias, Serviços Web Semânticos e Agentes de Software.

Outra contribuição é o framework UBICK, construído para estruturar as aplicações e proporcionar um ganho de produtividade nos seus desenvolvimentos baseado no reuso de componentes.

O estudo de caso desenvolvido para avaliar a abordagem proposta e apresentado nessa dissertação contribui para facilitar o entendimento e a transferência dos resultados produzidos nesta pesquisa. Este estudo, que consiste de uma aplicação no domínio da Educação Médica, foi desenvolvido para auxiliar o processo de ensino e aprendizagem no curso de Medicina da Universidade Federal de São Carlos (UFSCar).

1.4. Organização

Esta dissertação está organizada da seguinte forma:

Capítulo 2 apresenta uma visão geral sobre a Computação Ubíqua e propostas atuais para o desenvolvimento de software nesse novo paradigma computacional.

Capítulo 3 aborda os conceitos e técnicas usadas durante o desenvolvimento da abordagem proposta: Ontologias, Serviços Web Semânticos e Agentes de Software.

Capítulo 4 descreve o desenvolvimento do framework UBICK, construído para apoiar a abordagem proposta nessa dissertação. É apresentada sua Arquitetura, a Ontologia para descrição de contexto, e os seus componentes disponíveis para reuso no desenvolvimento de aplicações ubíquas.

Capítulo 5 apresenta a abordagem AdeSCoU, suas fases e disciplinas. São apresentadas as integrações das Ontologias, Serviços Web Semânticos e do UBICK, introduzidas pela abordagem, com o RUP.

Capítulo 6 apresenta um estudo de caso desenvolvido para avaliar a proposta deste trabalho. Este estudo de caso consiste do desenvolvimento de um ambiente para apoio do processo de ensino e aprendizagem no curso de Medicina da UFSCar.

Finalmente, o Capítulo 7 conclui este trabalho apresentando os resultados alcançados e contribuições, limitações e dificuldades encontradas, e possibilidades de trabalhos futuros.

Capítulo 2

Computação Ubíqua

O termo Computação Ubíqua refere-se a ambientes saturados de dispositivos computacionais e redes de comunicação sem fio, que se integram naturalmente à atividade humana. Segundo Mark Weiser, o pai da Computação Ubíqua, “as mais profundas tecnologias são as que desaparecem” [WEI91, WEI94]. Neste sentido a Computação Ubíqua pode ser considerada o oposto da Realidade Virtual. Enquanto na segunda o usuário penetra no mundo virtual criado pelos computadores, na primeira é a computação que penetra no mundo físico do usuário, construindo a ponte que liga esses dois mundos.

O avanço da capacidade de hardware, e de tecnologias chaves de software e de redes, estão tornando Computação Ubíqua uma realidade [PHA07]. Com isto, muitos ambientes foram desenvolvidos a fim de evidenciar as vantagens dessa nova era computacional. Entretanto, na grande maioria dos casos, esses ambientes possuem desvantagens como: desenvolvimento de forma manual e sob-demanda; falta de capacidade de expansão; e dificuldade de manutenção e evolução [HEL05].

Assim, ainda não é possível atingir o mundo previsto, há mais de uma década, pelos pioneiros da Computação Ubíqua [BEL07]. As dificuldades para o desenvolvimento desses ambientes são decorrentes das características não usuais da Computação Ubíqua como área de pesquisa. Isto porque, enquanto

a maioria das áreas de pesquisa em Ciência da Computação (e.g., Linguagens de Programação, Sistemas Distribuídos) é definida, em termos gerais, por problemas técnicos e é direcionada à solução de problemas do passado, a Computação Ubíqua abrange uma ampla variedade de áreas costuradas por uma visão comum e é direcionada às possibilidades do futuro [BEL07].

Tal cenário representa desafios de pesquisa em diferentes áreas, incluindo Redes de Computadores, Processamento de Sinais, Inteligência Artificial, e, particularmente, para a Engenharia de Software. Nesta área, tais desafios podem ser observados pela falta de metodologias para desenvolvimento, processos de software, abordagens para teste e técnicas de qualidade [SPI07].

Vários trabalhos descrevem os desafios para o desenvolvimento de software na Computação Ubíqua [SAT01, ABO99, KIN02, NIE04, END05, WAN05], no entanto, não existe uma classificação centralizada para definir projetos de software da Computação Ubíqua. Assim, nesta dissertação utiliza o critério apresentado em [SPI07]. Segundo este critério pode-se definir uma aplicação ubíqua, caso essa possua as seguintes características: Onipresença de Serviços *(Service omnipresence)*, que oferece aos usuários a sensação que os serviços computacionais os acompanham em seus movimentos; Invisibilidade *(Invisibility)*, habilidade de estar presente em objetos do cotidiano, sem que, do ponto de vista do usuário, exista a sensação de estar usando computadores; Sensibilidade à Contexto ou Ciência de Contexto *(Context sensitivity)*, a habilidade de coletar informações do ambiente onde a aplicação está sendo usada, denominadas contexto. O contexto é definido como os atributos que podem ser utilizados para caracterizar o ambiente de uso de uma

aplicação, incluindo seu usuário e a própria [DEY01]; Capacidade de Adaptação (*Adaptable behavior*), capacidade de adaptar os serviços oferecidos, dinamicamente, conforme o contexto; Captura da Experiência *(Experience capture)*, habilidade de capturar, registrar e, posteriormente, usar as experiências de uso da aplicação; Descoberta de Serviços *(Service discovery)*, descobrir pró-ativamente serviços conforme o contexto de uso; Composição de Serviços *(Function composition)*, baseado nos serviços, deve-se criar novos serviços, requeridos pelo contexto; Interoperabilidade (*Spontaneous interoperability*), habilidade de modificar seus padrões, permitindo a continuidade do uso da aplicação conforme o movimento do usuário; Heterogeneidade (*Heterogeneity of devices*), oferece mobilidade da aplicação entre dispositivos; Tolerância a Falhas *(Fault tolerance)*, habilidade de adaptar-se durante falhas na execução.

2.1. Engenharia de Software na Computação Ubíqua

Com o objetivo de correlacionar a proposta dessa dissertação, essa seção apresenta uma revisão sistemática da literatura[1] [KIT04] em relação à Engenharia de Software (ES) na Computação Ubíqua.

A primeira atividade dessa revisão sistemática foi definir a seguinte pergunta: quais são as técnicas da ES propostas na Computação Ubíqua? Os critérios para seleção de fontes de consulta que respondessem essa pergunta foram: disponibilidade de artigos via Web; existência de mecanismos de busca através de palavras-chave; e a presença de pelo menos uma das características apresentadas seção anterior. Para escolha das palavras-chave,

[1] *Uma revisão sistemática significa identificar, avaliar e interpretar as pesquisas mais relevantes para uma pesquisa, área ou fenômeno particular. Esta abordagem, ao contrário de abordagens sob-demanda, é baseada num método científico e segue um protocolo para conduzir uma revisão formal.*

combinaram-se palavras sobre Computação Ubíqua com palavras para Engenharia de Software. As palavras do primeiro tipo foram: *ubiquitous* e *pervasive*. As expressões do segundo tipo foram: *software engineering*, *software development* e *software process*. Foram considerados os seguintes critérios de inclusão: os artigos devem estar disponíveis nos mecanismos de busca *ACM Portal*, *IEEE Xplore*, *Elsevier* e *SpringerLink*; os artigos devem apresentar textos completos em formato eletrônico; os artigos devem ser escritos em inglês; o artigo deve ser recente (posterior a 2005). Foram excluídos os artigos que tratavam de processos para integração de hardware (e.g., [KRI05]) e os artigos que apresentassem propostas dependentes de domínio (e.g., [BAR07]). Para seleção, o autor dessa dissertação leu os resumos e as introduções dos artigos e, baseando-se nos critérios de inclusão e exclusão, selecinou-os, ou não, para maiores análises. De acordo com as consultas foram encontrados 62 artigos que atenderam aos critérios de inclusão. A Tabela 2.1 apresenta 12 trabalhos mais representativos em relação à abordagem proposta.

Tabela 2.1. Abordagens para o Desenvolvimento de Software na Computação Ubíqua.

Autor Principal	Tipo	Características
Balasubramanian	Framework	Ciência de Contexto, Onipresença de Serviços, Capacidade de Adaptação, *Invisibility*
Soroker	Framework	Capacidade de Adaptação, Ciência de Contexto
Wu	Framework	Onipresença de Serviços, Capacidade de Adaptação e Ciência de Contexto
Aitenbichler	*Middleware*	Onipresença de Serviços, Interoperabilidade, Heterogeneidade, Capacidade de Adaptação
Carton	Metodologia/ Processo	Ciência de Contexto, Capacidade de Adaptação
Pham	Metodologia/ Processo	Ciência de Contexto, Capacidade de Adaptação

Fernandes	Metodologia/ Processo	Capacidade de Adaptação, Ciência de Contexto
Bulcão Neto	Metodologia/ Processo	Ciência de Contexto
Syukur	Metodologia/ Processo	Ciência de Contexto
Riva	Padrões	Ciência de Contexto, Interoperabilidade
Min	Modelo de Programação	Capacidade de Adaptação, Ciência de Contexto
Devdatta	Modelo de Programação	Ciência de Contexto

Os artigos selecionados apresentam diferentes propostas para o desenvolvimento de aplicações ubíquas, destacando-se: frameworks, middlewares, ferramentas, modelos de programação, metodologias e padrões de projeto. Alguns projetos atendem as características da Computação Ubíqua disponibilizando componentes previamente implementados (frameworks, middlewares e ferramentas) para que sejam usados durante o desenvolvimento de novas aplicações, outros, visam obter tais características através do processo de desenvolvimento (modelos de programação, metodologias e padrões de projeto).

O trabalho de Balasubramanian [BAL06] apresenta um framework para rápida prototipagem de aplicações ubíquas. Este framework apresenta componentes para os clientes e para servidores. Da mesma forma o trabalho de Soroker [SOR07] apresenta um framework para prototipagem de aplicações ubíquas, denominado *Pegboard*.

O trabalho de Wu [WU08] apresenta um framework para facilitar para o desenvolvimento de aplicações ubíquas, que tenham como características adaptabilidade e onipresença de serviços. Este framework é formado pelos módulos: monitor de eventos, que verifica modificações no contexto; aplicação

de regras, que interpreta as modificações nos eventos; e interface com banco de dados, que permite a aplicações ubíquas.

O trabalho de Aitenbichler [AIT07] apresenta um middleware de baixa complexidade, denominado Mundo Core, para lidar com as características de heterogeneidade e adaptação na Computação Ubíqua.

O trabalho de Carton [CAR07] apresenta um processo de que combina *Apect-Oriented Software Development (AOSD)* e *Model-Driven Development (MDD)*. Este processo argumenta que o AOSD facilita a divisão das características adaptáveis (e.g., dispositivo, localização, usuário), a fim de desenvolver componentes pouco acoplados, que possam ser reutilizados.

O MDD é usado também no trabalho de Pham [PHA07] e Fernandes [FER07], nos quais este tipo de metodologia, facilita o desenvolvimento de aplicações independentes de plataforma a fim de facilitar a manutenção das mesmas.

O trabalho de Bulcão Neto [BUL07] é um processo de desenvolvimento de aplicações sensíveis ao contexto, denominado *Process for Ontological Context-aware Applications (POCAp)*. Esse método baseia-se em Ontologias [GUI05] para descrição de contexto.

O trabalho de Suyur [SUY06] apresenta uma metodologia, denominada MHS, para análise e projeto de aplicações cientes de contexto. Esta metodologia é dividida em três fases: análise, projeto e implementação; contudo, a implementação não é detalhada pela MHS. A fase de análise é dividida nas etapas: captura de requisitos; refinamento de escopo, que visa especificar os requisitos capturados; e construção de elementos do sistema, que visa modelar elementos da aplicação ubíqua (e.g., usuários, ambientes e

contextos). A fase de projeto é formada por duas subfases, denominadas: projeto conceitual e projeto funcional.

O trabalho de Riva [RIV06] apresenta quatro padrões de desenvolvimento. Os padrões *Flyweight* e *Hybrid Mediator-observer*, são adaptados dos padrões de projeto GOF. Além disso, este trabalho propõe os padrões *Flexible Context Processing* e *Enactor*. O primeiro provê reuso de elementos do contexto, o segundo oferecer controle para apoiar o controle e monitoramento de contexto.

O trabalho de Min [MIN07] apresenta um modelo de programação denominado *Isotype*. Este novo modelo de programação amplia o conceito de objetos, transformando-os em uma agregação de atributos e uma série de comportamentos influenciados por diferentes contextos. Cada um desses novos objetos representa a informação sobre o ambiente, e o comportamento que este objeto deve ter de acordo com o contexto.

O trabalho de Devdatta [KUL07] apresenta um modelo de programação que permite um desenvolvimento rápido de aplicações cientes de contexto a partir de especificações em alto nível. Esta abordagem utiliza-se do paradigma de programação gerativo, e tem como idéia central que um ambiente execução gere a aplicação ubíqua ao integrar políticas com um conjunto de componentes e serviços de um middleware.

2.2. Conclusões

De maneira geral, as abordagens procuram oferecer ao Engenheiro de Software uma implementação inicial, a fim de diminuir a complexidade do desenvolvimento. Entretanto, são poucos trabalhos sistematizam o Processo de Desenvolvimento de Software (PDS) na Computação Ubíqua.

Visando contribuir para esse melhoria do PDS na Computação Ubíqua, esta dissertação propõe uma abordagem de desenvolvimento baseada no RUP, modificado pelo emprego de Ontologias, Serviços Web Semânticos, e Agentes de Software, e com apoio computacional de um framework que organiza o reuso baseado em componentes de software. Para facilitar o entendimento da abordagem proposta, o próximo capítulo apresenta uma visão geral sobre Ontologias, Serviços Web Semânticos e Agentes de Software.

Capítulo 3

Conceitos e Técnicas da Abordagem

Para desenvolvimento da *Abordagem para o Desenvolvimento de Software na Computação Ubíqua (AdeSCoU),* foi construído um framework baseado nos seguintes conceitos e técnicas: Ontologias; Serviços Web Semânticos; e Agentes de Software. A fim de facilitar o entendimento da seqüência dessa dissertação, as próximas seções oferecem uma visão geral sobre esses conceitos e técnicas.

3.1. Ontologias

Para Ciência da Computação, as Ontologias dizem respeito a um modelo de dados, que representa um domínio e pode ser utilizado para inferir conhecimento sobre objetos e relações nesse domínio [GUI05].

Os elementos de uma Ontologia são [GUI05]: classes ou conceitos, grupos, conjuntos ou coleções abstratas de objetos, podendo conter indivíduos, outras classes ou ambos. (e.g, Pessoa, classe de todas as pessoas); indivíduos ou instâncias, componentes básicos de uma Ontologia, podendo incluir objetos concretos (e.g., uma pessoa) e abstratos (e.g., uma palavra); atributos, propriedades que caracterizam um objeto; e relações são atributos que descrevem como os objetos se relacionam.

Dentre as várias linguagens propostas para representação de Ontologias, a *Ontology Web Language (OWL)* [MCG04] se destaca por ser um

padrão recomendado pelo *WWW Consortiun (W3C)* [W3C04]. OWL adiciona as seguintes capacidades às Ontologias: habilidade de ser distribuído através de várias aplicações; escalabilidade; compatibilidade com padrões Web para acessibilidade e internacionalização; e extensibilidade.

Na Computação Ubíqua, as Ontologias vêm sendo utilizadas majoritariamente para descrição de contexto [CHE04, FOR06, BUL06]. Entretanto, existem também pesquisas que as utilizam para outros fins: em [CHR05] são utilizadas para definir uma linguagem comum para comunicação e colaboração entre dispositivos de um ambiente ubíquo; em [PAN05] são utilizadas para gerenciamento dinâmico de recursos (e.g., sensores, atuadores) em ambientes ubíquos; finalmente, em [KIM06] são usadas para facilitar reconfiguração dinâmica.

3.2. Serviços Web Semânticos

Serviços Web tradicionais possibilitam que novas aplicações interajam com outras já existentes e que aplicações desenvolvidas em plataformas distintas se tornem compatíveis [PAP03].

A Figura 3.1 ilustra a evolução da Web e dos Serviços Web. Com a introdução de linguagens semânticas, baseadas na Web Semântica [BER01], os Serviços Web ganharam poder de descrição suficiente para o computador determinar os significados abstraídos nesse serviço [OAS], tornando-os Serviços Web Semânticos[2]. Tais linguagens devem oferecer recursos que permitam a descoberta dos serviços necessários, suas invocações e, caso seja necessário compor vários para atingir um objetivo.

[2] *Nesta dissertação o termo Serviços Web Semânticos refere-se ao uso de tecnologias da Web Semântica para a descrição, a descoberta e a composição de Serviços Web.*

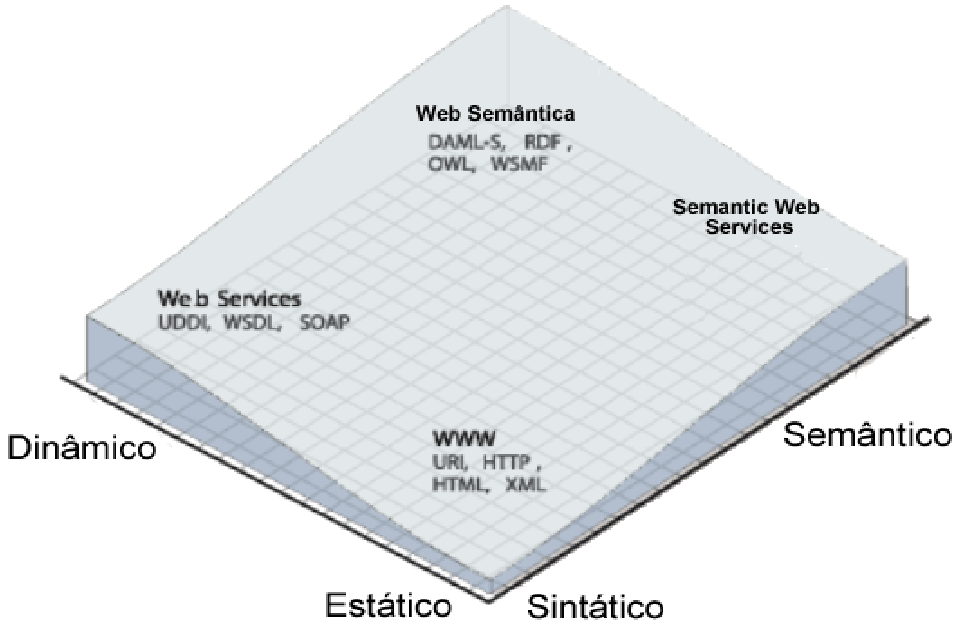

Figura 3.1. Evolução dos Serviços Web [REF]

Dentre as linguagens para especificação de Serviços Web Semânticos, destaca-se a *Ontology Web Language for Service (OWL-S)*, proposta como padrão do *W3C*. OWL-S é uma Ontologia baseada em OWL que consiste de uma linguagem de marcação descrição das propriedades e capacidades de Serviços Web de modo não ambíguo e interpretável por computadores, automatizando a descoberta, a execução e a composição desses serviços [MAR03].

OWL-S é formada por três partes: *Service Profile*, *Service Model*, e *Service Grounding*. Service Profile é uma descrição de alto-nível do provedor e das funcionalidades do serviço. Service Profile descreve as seguintes características: informações de contato de um determinado provedor/serviço; atributos para categorização do serviço oferecido; e as representações funcionais do serviço em forma de entradas e saídas. Service Model define como o serviço opera, e possui a subclasse ProcessModel, que representa os serviços através de processos. Cada processo é visto como a transformação

de um conjunto de entradas para um conjunto de saídas, e como uma transição no contexto de um estado para outro. Service Grounding especifica os detalhes de como acessar o serviço. É a única parte de um serviço OWL-S num nível concreto de especificação, sendo relacionado ao *Web Services Description Language (WSDL)* [W3C01].

3.3. Agentes de Software

Agentes de Software podem ser definidos como sistemas computacionais, que possuem todas, ou grande parte, das seguintes características [ZAM04, FRA96]: posicionamento (*situatedness*), os agentes devem conhecer o ambiente em que está inserido e executar ações contextualizadas para modificar este ambiente; autonomia (*autonomy*), os agentes devem agir sem a intervenção direta de usuários ou de outros agentes; pró-atividade (*pro-activeness*), os agentes devem agir oportunamente por iniciativa própria de acordo com seus objetivos; sociabilidade (*sociability*), os agentes devem interagir com outras entidades do ambiente para resolver seus problemas ou auxiliá-las em suas atividades; adaptabilidade (*adaptiveness*), os agentes podem mudar o seu comportamento de acordo com experiências passadas; receptividade (*responsiveness*), os agentes devem responder adequadamente a mudanças que ocorram no ambiente; e mobilidade (*mobility*), os agentes podem transportar-se de um computador para outro.

Os Agentes de Software desenvolvidos nesse trabalho possuem como característica a mobilidade, por isso são denominados Agentes Móveis [SER98]. Segundo [LAN99] as principais vantagens no uso de Agentes Móveis são: *redução do tráfego da rede*, pois permite despachar tarefas e mover o processamento para o local onde os dados estão armazenados; *ocultar a*

latência da rede, pois oferece uma solução para controle de aplicações críticas, mesmo em redes grandes, já que podem ser despachados pelo controlador central para realizarem suas tarefas localmente; *encapsular o protocolo da rede*, pois oferece uma camada de abstração sobre a camada de rede; *executar de forma assíncrona e autônoma*, pois as tarefas podem ser embutidas em agentes ser despachados pela rede, de forma autônoma e independente da criação de processo; *adaptar-se dinamicamente*, pois possuem a habilidade de perceber mudanças no ambiente de execução e reagir autonomamente; *ser Independente de plataforma*, pois são dependentes somente do seu ambiente de execução; e *robustez e tolerância a falhas*, pois habilidade de reagirem dinamicamente, permite que, na eminência de situações desfavoráveis, todos os agentes em execução num computador possam ser despachados e continuar suas tarefas em outro computador da rede.

Muitos pesquisadores têm apontado para o paradigma multi-agentes como adequado ao desenvolvimento de aplicações complexas [ZAM04, JEN99]. Sistemas Multi-Agentes (SMA) permitem modelar o comportamento de um conjunto de entidades inteligentes e organizadas de acordo com leis do tipo social. Estas entidades, ou agentes, dispõem de autonomia e estão imersos num ambiente com o qual necessitam interagir. A necessidade desta interação resulta da existência de dependência entre as tarefas a executar pelos vários agentes, da obrigatoriedade na satisfação de restrições globais, ou ainda da incapacidade de resolução de uma tarefa individualmente [JEN98]. As entidades fundamentais que compõem um SMA são: a) Agentes, entidades autocontidas capazes de interagir, observar e agir, autonomamente em relação

ao ambiente; b) Recursos ou Objetos, entidades físicas ou de informação; c) Ambientes, entidades lógicas ou físicas que cercam o agente e oferecem condições para que os agentes existam e realizem suas tarefas; e d) Organizações agrupam os agentes de um SMA. Uma organização pode definir um conjunto de axiomas o qual os agentes devem obedecer. Um axioma é uma regra, lei ou princípio estabelecido. Uma organização também define os papéis que devem ser exercidos pelos agentes dentro dela e os papéis que devem ser exercidos pelos objetos. Como os agentes, as organizações interagem com outras entidades enviando e recebendo mensagens.

Na Computação Ubíqua, os Agentes de Software podem auxiliar em diferentes tarefas como, por exemplo, onipresença de serviços, capacidade de adaptação, qualidade de serviço [CAR02]. No trabalho de Lee [LEE07], Agentes de Software monitoram contexto, adaptam conteúdo e acessam serviços. O trabalho de Soldatos [SOL07] apresenta um middleware que facilita o uso de serviços ubíquos, oferecendo um mecanismo ciente de contexto, para controle para sensores e atuadores. Além disso, Agentes de Software podem ainda ser empregados para a utilização de Ontologias [JAD] e Serviços Web Semânticos [CHA06].

3.4. Conclusões

Este capítulo apresentou os conceitos e tecnologias envolvidas na abordagem proposta. Ontologias, Serviços Web Semânticos foram discutidos brevemente a fim de auxiliar no entendimento da abordagem proposta nesse trabalho. Ontologias facilitam a representação de um domínio e pode ser utilizada para inferir sobre objetos e relações nesse domínio. Serviços Web Semânticos adicionam uma descrição semântica aos Serviços Web Tradicionais, a fim de

automatizar a descoberta, a execução e a composição de componentes distribuídos. Finalmente, foram apresentados os Agentes de Software, que oferecem a abstração necessária o desenvolvimento de aplicações adaptáveis, tolerantes à falhas e seguras, além de facilitar a utilização de Serviços Web Semânticos.

Capítulo 4

Framework UBICK

Este capítulo apresenta um framework, denominado *Ubiquitous Computing frameworK (UBICK),* que atua como mecanismo computacional de apoio da abordagem AdeSCoU nas fases de projeto e implementação das aplicações ubíquas. O UBICK, evolução dos trabalhos de Claudino [CLA05] e Forte [FOR07], define uma estrutura para organizar e desenvolver aplicações ubíquas, incluindo modelos e componentes para reuso.

Segue-se um relato sobre os requisitos, modelagem e implementação da versão atual do UBICK.

4.1. Requisitos

Os requisitos do UBICK basearam-se principalmente nas características da Computação Ubíqua sob o prisma da Engenharia de Software, apresentadas na Seção 2.1.

Assim, foram identificados os principais Atores[3] e Casos de Uso [OMG04], ilustrados na Figura 4.1. O ator denominado Dispositivo de Apresentação, que representa qualquer dispositivo capaz de apresentar informações (e.g., Celulares, Palms, PCs), está relacionado ao Caso de Uso Acessar Conteúdo, sendo necessário usar o contexto, para conhecer suas preferências e necessidades para adaptar conteúdo. Esse ator relaciona-se, da

[3] *Ator: algo que interage com a aplicação, mas sobre o qual a aplicação não tem controle*

mesma forma, nas ocasiões em que é necessário utilizar o contexto para Acessar Serviços Web Semânticos.

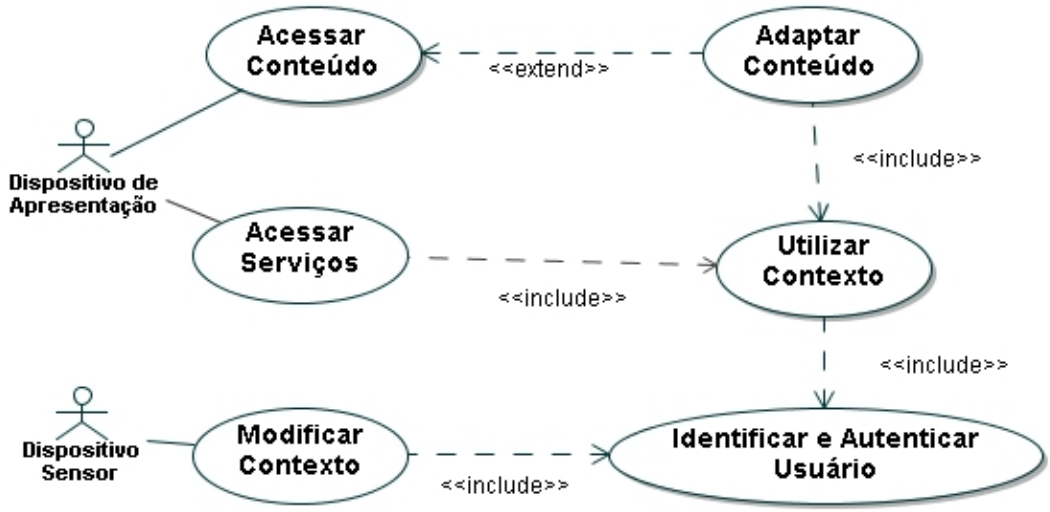

Figura 4.1. Principais casos de usos identificados

O ator Dispositivo Sensor representa dispositivos capazes de monitorar as informações relativas ao contexto de uso da aplicação (e.g., GPS, câmera, microfone). Esse ator relaciona-se com o caso de uso Modificar Contexto, para tal, também é necessário Identificar e Autenticar Usuário, a fim de relacionar a modificação de contexto com o usuário. Da mesma forma procedeu-se para outros requisitos, representando em modelos de casos de uso com suas respectivas documentações.

Esses modelos iniciais possibilitaram elicitar, ou seja, tornar mais claro, os seguintes requisitos funcionais do UBICK:

a) Ciência de Contexto: o UBICK deve permitir que Engenheiros de Software utilizem os atributos do ambiente computacional, pessoal e físico para oferecer conteúdos e serviços customizados para o usuário;

b) Adaptabilidade: o UBICK deve permitir que uma aplicação organize sua operação de acordo com os atributos do contexto;

c) Adaptação de Conteúdo: o UBICK deve permitir acesso ubíquo à informação, requisito fundamental para atingir o mundo previsto pelos pioneiros da Computação Ubíqua [FAL07], uma vez que, apesar do uso crescente dos dispositivos móveis, a criação de conteúdo ainda é majoritariamente direcionada aos PCs; e

d) Onipresença, Descoberta e Composição de Serviços Web Semânticos: o UBICK deve permitir acesso onipresente aos serviços da aplicação (e.g., acesso à conta bancária). Considerando o contexto, esses serviços devem ser descobertos, e, eventualmente, compostos para realização de tarefas complexas.

Além desses, foram identificados os seguintes requisitos não funcionais[4], que foram tratados no desenvolvimento do UBICK:

a) Interoperabilidade e Heterogeneidade: as funções do UBICK devem ser interoperável em todos seus níveis (e.g. dispositivos, serviços, hardware, software), já que ambientes da Computação Ubíqua são formados por uma grande quantidade de dispositivos heterogêneos;

b) Segurança e Privacidade: as funções do UBICK devem ser realizadas de forma segura, uma vez que informações confidenciais, como as manipuladas em transações para alteração e uso do contexto, trafegam pelos diferentes elementos da rede ;

c) Tolerância à Falhas: as funções do UBICK devem ser tolerantes a falhas, já que a definição da Computação Ubíqua torna os usuários dependentes dessa

[4] *Requisitos não funcionais são qualidades globais de um software, que expressam como devem feitas realizadas as funções da aplicação. Em geral se relacionam com padrões de qualidade como confiabilidade, performance, robustez, manutenibilidade, usabilidade, desempenho e custo.*

aplicação, de tal maneira que a aplicação deve ser confiável também em relação ao seu desempenho.

Os requisitos Invisibilidade e Captura de Experiência não foram considerados, pois podem ser tratados durante o processo de desenvolvimento.

4.2. Análise

Prosseguindo com o desenvolvimento do UBICK, seus requisitos foram analisados e refinados resultando na: criação de sua arquitetura; especificação de uma Ontologia para descrição de contexto; e especificação inicial dos componentes do UBICK.

4.2.1. Arquitetura do UBICK

A arquitetura do UBICK baseou-se no modelo computacional Cliente e Servidor. Cada Cliente pode enviar requisições para algum dos Servidores conectados e esperar pela resposta. Por sua vez, os Servidores podem aceitar tais requisições, processá-las e retornar o resultado para o Cliente.

A Figura 4.2 apresenta a arquitetura do UBICK, com seus principais elementos: Clientes, Servidores e Serviços Web Semânticos. Um usuário acessa uma rede (e.g., Internet, Intranet) como um Cliente, que é capaz de acessar conteúdos e serviços, monitorar o contexto de uso da aplicação e armazenar dados para que sejam posteriormente enviados ao Servidor.

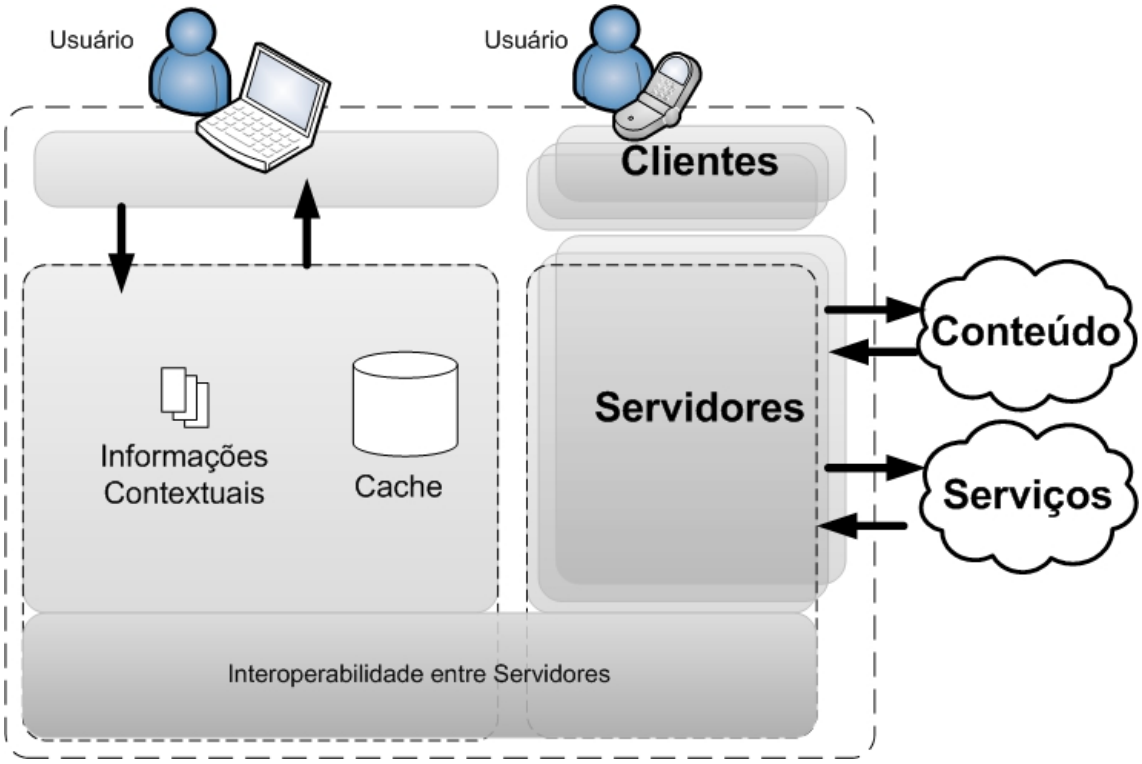

Figura 4.2. Esboço de Arquitetura do UBICK

Servidores, por sua vez, desempenham tarefas que necessitam de maior poder computacional, tais como: (a) inferências sobre informações contextuais para organização da aplicação e uso de serviços; (b) políticas de segurança; (c) descoberta e composição de serviços acesso; (d) *Cache* dos conteúdos acessados para aumento do desempenho; e (e) compartilhamento de Serviços e Informações Contextuais para aumentar a tolerância à falhas da arquitetura.

4.2.2. Descrição do Contexto

Ontologias foram usadas para descrever o contexto de uso das aplicações. No modelo de Ontologias da Figura 4.3, as classes *UserProfile*, *DeviceProfile*, *NetworkProfile*, *ContentProfile* e *ServiceProfile*, representam, respectivamente, os perfis de usuário, dispositivo, rede, conteúdo e serviços. Na classe UserProfile as propriedades *id* e *name* são utilizadas pelo servidor para identificar um usuário. A propriedade *hasPreferences* indica as preferências do usuário em relação às adaptações (e.g., receber apenas textos em português).

A propriedade *hasDevice* indica o tipo de dispositivo de acesso. Na classe DeviceProfile as propriedades *brand*, m*odel*, *hasDisplay*, e *hasSensor* descrevem a marca, o modelo, as características da tela do dispositivo móvel e os sensores que o dispositivo possui. A classe NetworkProfile tem as propriedades *latency* e *protocol* para descrever a latência e o protocolo utilizado pela rede. A classe ContenProfile tem as propriedades: *id*, que indica o caminho para o conteúdo; *format*, que define o formato do conteúdo (e.g., HTML); e *hasPart*, uma vez que um conteúdo pode ser formado por outros conteúdos. Na classe ServiceProfile, que representa os Serviços Web Semânticos têm-se as propriedades: *hasInput*, para descrever os tipos das entradas do serviços; e *hasOutput*, para descrever os tipos das saídas dos serviços.

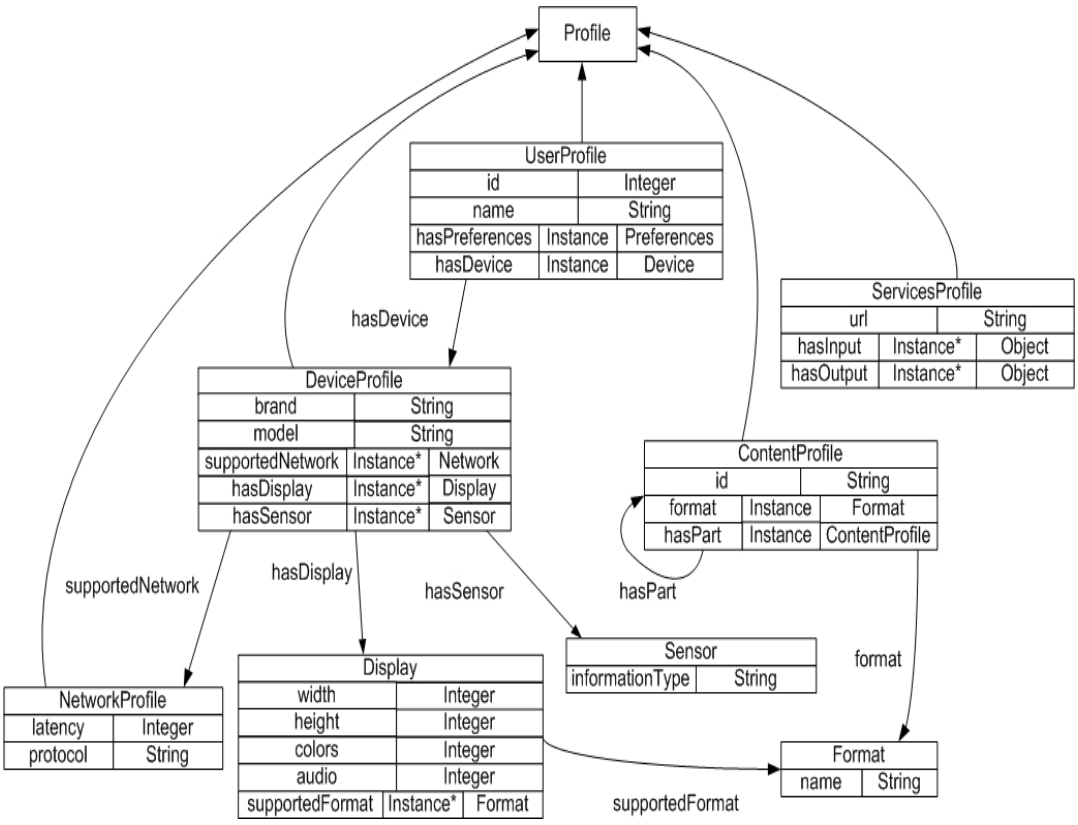

Figura 4.3. Ontologia para Descrição do Contexto

Através da Ontologia definiu-se um conjunto mínimo de conceitos, propriedades e relações para uma aplicação da Computação Ubíqua. Entretanto, para cada domínio específico essa Ontologia pode ser estendida, como a do estudo de caso apresentado no Capítulo 6.

4.2.3. Agentes de Software e Serviços Web Semânticos

Prosseguindo com a modelagem, identificou-se que Agentes de Software podem facilitar o uso de Serviços Web Semânticos [CHA06], monitoramento de contexto, segurança e adaptabilidade [SOL07].

Utilizou-se a *Multi-Agents System Modeling Language (MAS-ML)* [TOR04, TOR07], para especificação dos Agentes de Software. Essa linguagem é formada por um conjunto de elementos que representam diferentes aspectos de aplicações multi-agentes, possibilitando que os Engenheiros de Software incorporem engenharia de aplicação multi-agentes, nos processos de análise e modelagem de aplicações.

A Figura 4.4 apresenta um modelo com os agentes de software, suas organizações, e relacionamentos. Assim, foram definidas as Organizações: *Client* e *Server*, que representam clientes e servidores na arquitetura. Continuando, foram definidos os Agentes de Software *ManagerAgent (MA)*, *ContextAgentServer (CAS)*, *ContextAgentClient (CAC)*, *ServicesAgent (SA)*, *ContentAccessAgent (CAA)*, *SharingAgent (ShA)* e *BrokerAgent (BA)*. Finalmente, foram definidos os objetos *Context*, *ContentStore* e *Cache*.

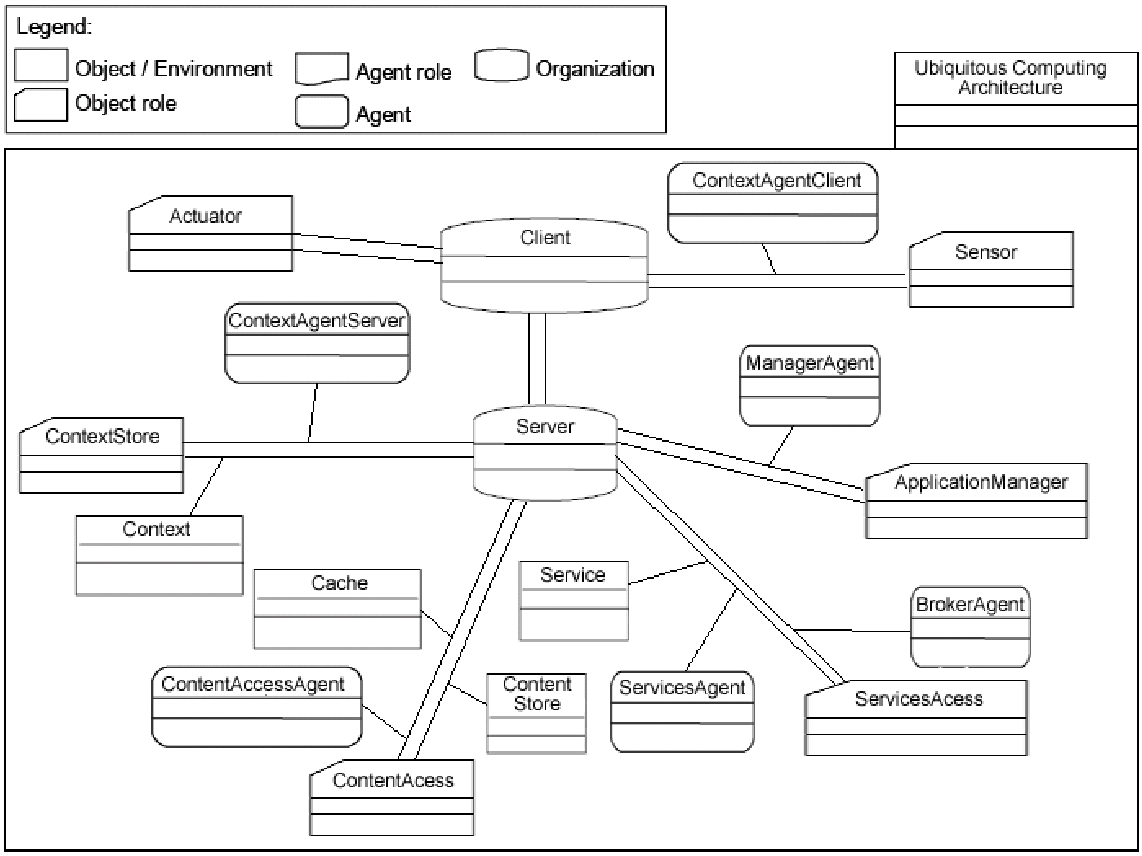

Figura 4.4. Agentes de Software

A Figura 4.5 apresenta o cenário para atualização de contexto. Inicialmente, o ContextAgentClient (CAC), requisita o contexto (e.g., preferencias do usuário, características do dispositivo, situação da rede) a um sensor. Caso haja mudanças, o CAC as envia ao ContextAgentServer (CAS), que atualiza o contexto da aplicação.

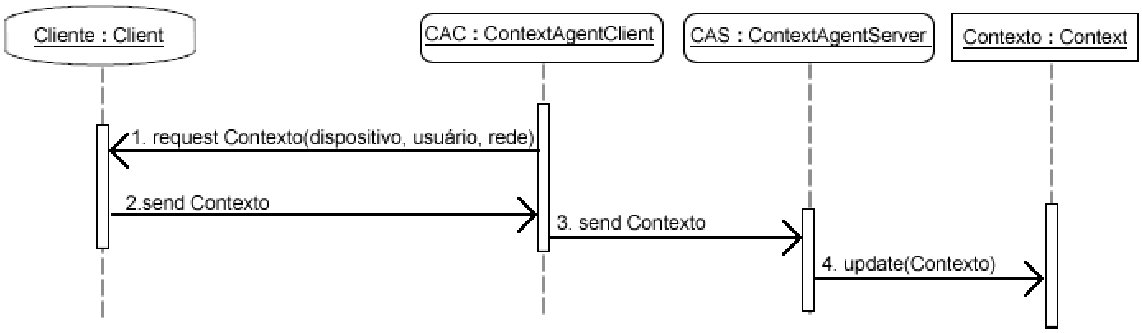

Figura 4.5. Cenário da Atualização de Contexto

A Figura 4.6 apresenta o cenário para recuperação de contexto. Inicialmente, uma requisição de um cliente é interceptada pelo ManagerAgent (MA). Na seqüência, o MA requisita ao Servidor que identifique e crie um BrokerAgent (BA) para o usuário. Em seguida, o Servidor envia o BA criado para o MA, que se comunica com este BA para verificar qual é o contexto de uso da aplicação em relação a esse usuário.

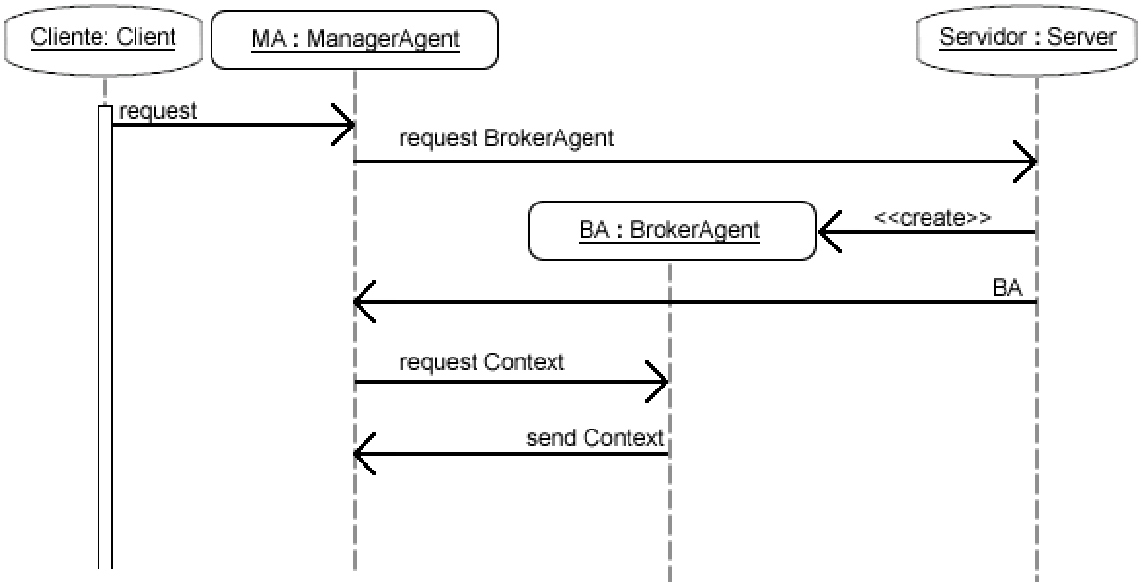

Figura 4.6. Cenário da recuperação do Contexto

A Figura 4.7 apresenta o cenário para uso de Serviços Web Semânticos. Após uma requisição por um serviço, o MA recupera e realiza inferências sobre o contexto. Conforme as inferências realizadas o MA envia ao ServiceAgent (SA) uma requisição por um Serviço. O SA encontra, compõe e invoca os Serviços Web Semânticos necessários para satisfazer essa requisição. Finalmente, o SA envia a resposta desse serviço ao MA, que por sua vez a envia ao Cliente.

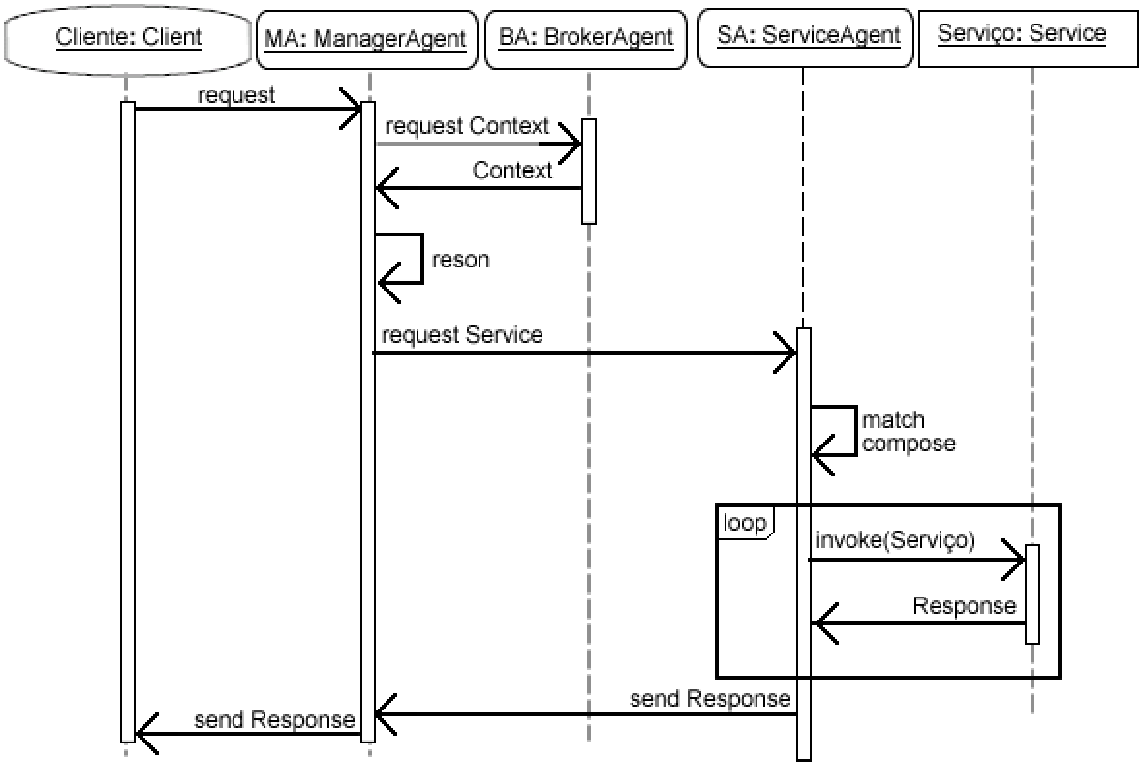

Figura 4.7. Cenário do uso de Serviços Web Semânticos

A Figura 4.8 apresenta o cenário de acesso aos Conteúdos. Após uma requisição por conteúdo o MA envia essa requisição ao ContentAcessAgent (CAA), que verifica que no Cache não há uma versão do conteúdo que atenda as necessidades do contexto. Assim, o CAA recupera o conteúdo do seu local de armazenamento (e.g., servidor Web, banco de dados). Em seguida, o CAA envia ao Service Agent (SA) uma requisição por um serviço de adaptação de conteúdo. O SA encontra, compõe e invoca os Serviços Web Semânticos. Finalmente, o conteúdo adaptado é entregue ao CAA, que atualiza o Cache e envia este conteúdo ao MA.

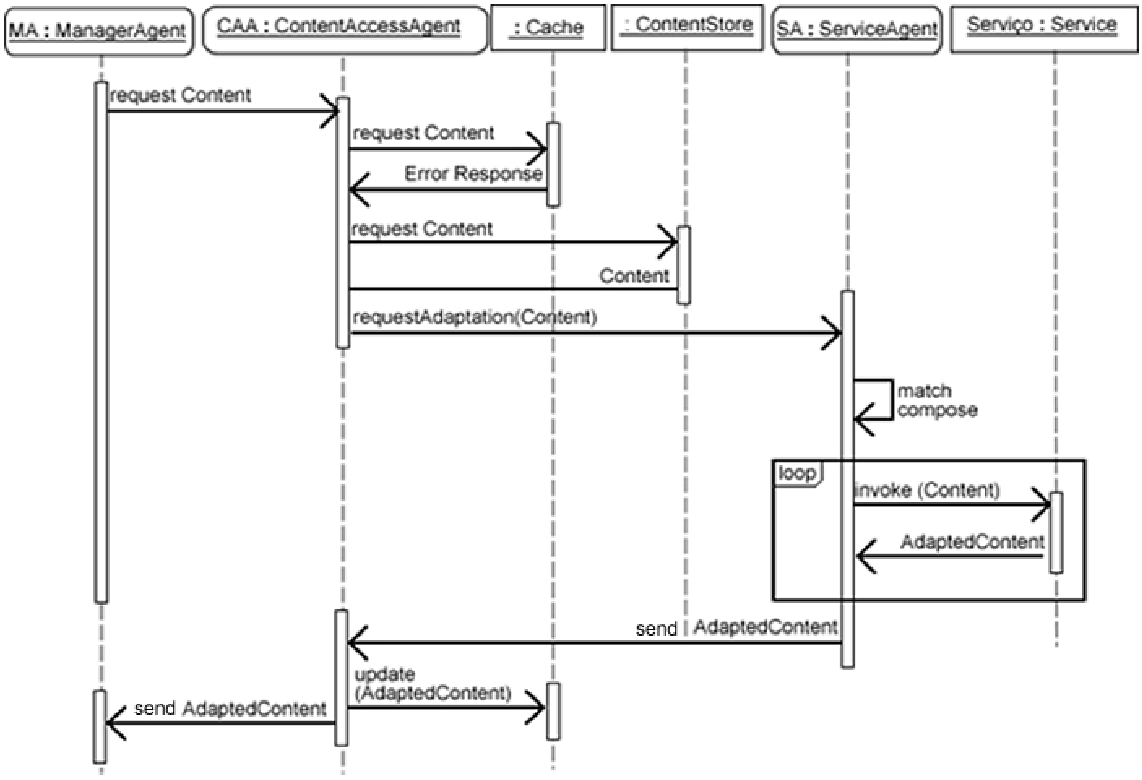

Figura 4.8. Cenário de acesso aos Conteúdos

Para realizar uma adaptação de conteúdo mais complexa, muitas vezes é necessário empregar mais de um Serviço Web Semântico de adaptação. Assim, foram identificados dois casos de composição. O primeiro denominado seqüencial ocorre quando um serviço não disponível é substituído por outros organizados seqüencialmente, ou seja, a saída de um serviço representa a entrada do próximo. O segundo denominado paralelo ocorre quando um conteúdo complexo deve ser adaptado. Essa divisão por processos paralelos torna mais simples o algoritmo e requer a arquitetura de adaptação certifique que todas as adaptações foram completadas.

A Figura 4.9 apresenta um exemplo da composição seqüencial para adaptação de conteúdos no formato *Portable Document Format (PDF)* para a linguagem *Wireless Markup Language (WML)*, de um determinado dispositivo, através dos Serviços Web Semânticos A, B, C, D e F. Uma vez que nenhum

desses serviços é capaz de realizar essa adaptação, tais serviços devem ser compostos. Nesse exemplo, duas composições, ABC e DF, atendem à adaptação necessária.

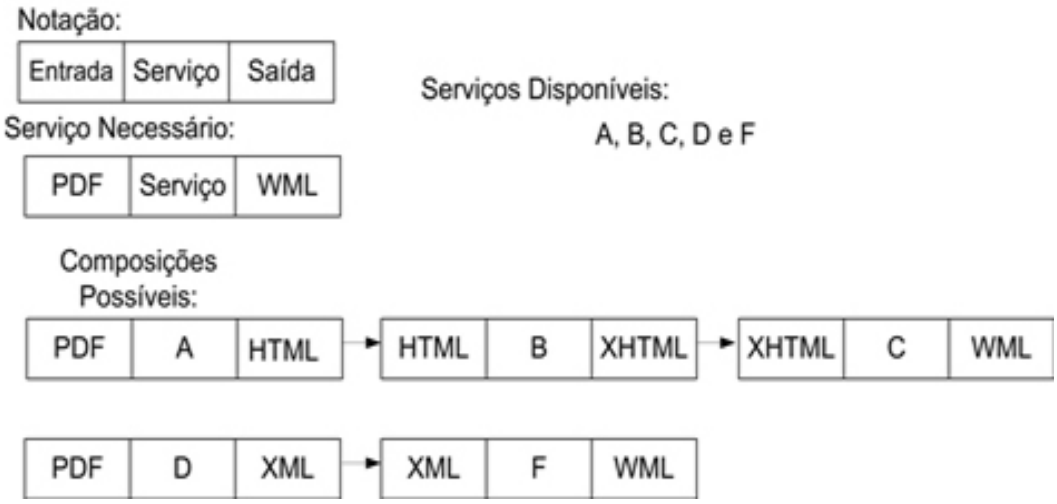

Figura 4.9. Composição Seqüencial de Serviços Web Semânticos

A Figura 4.10 ilustra a composição paralela, para adaptação de uma página Web. Nesse caso, a página deve ter sua linguagem de marcação traduzida, e, suas imagens e seus vídeos transcodificados. Como ilustrado cada adaptação é independente das outras.

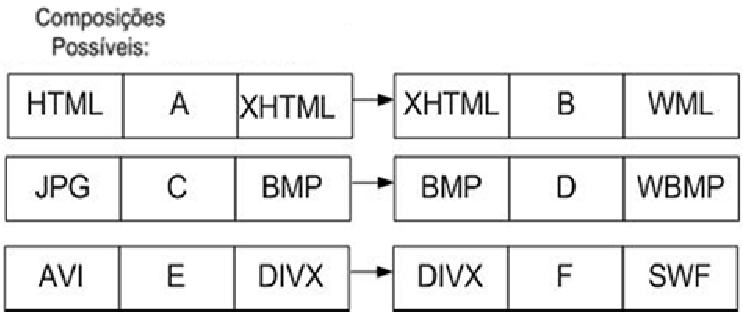

Figura 4.10. Composição Paralela de Serviços Web Semânticos

Em outros casospodem ser necessárias interações entre os servidores, como apresentado na Figura 4.11. Assim, quando o contexto de um usuário não está disponível no servidor no qual está conectado, ou quando um serviço

necessário não está disponível, um *SharingAgent* é instanciado, e seus clones são enviados aos Servidores conhecidos pelo servidor local. Nesses servidores, os clones comunicam-se com ManagerAgent local para encontrar o recurso (contexto do usuário ou serviço), que está faltando. Posteriormente, cada um dos clones retorna para o Servidor de origem com o recurso ou uma mensagem de erro.

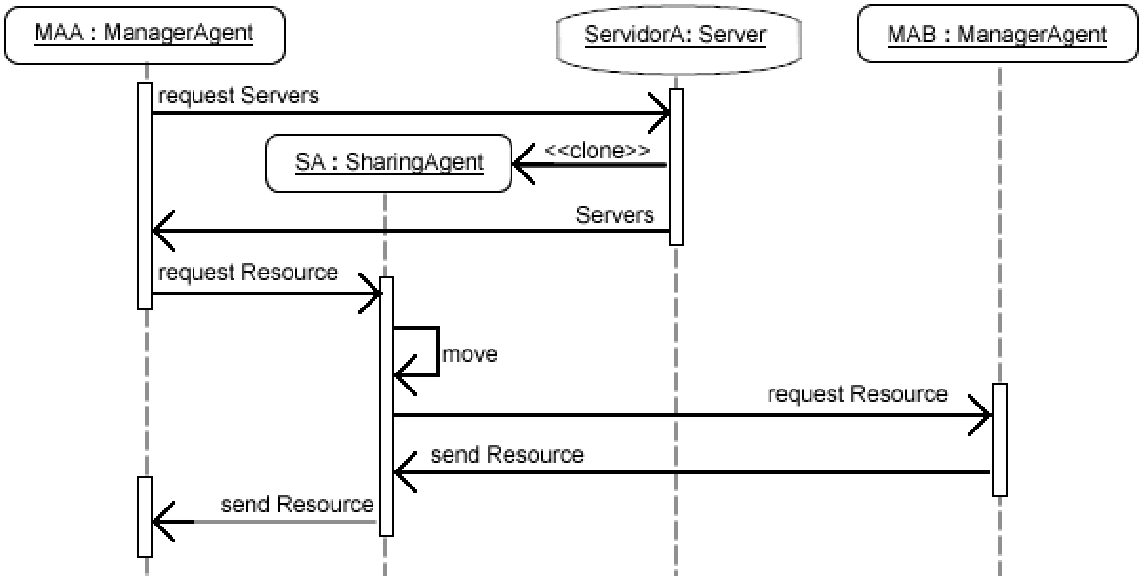

Figura 4.11. Cenário da comunicação entre servidores

4.3. Projeto

Para projeto do UBICK foram utilizados os seguintes frameworks: JADE [BEL08, JAD] para construção dos Agentes de Software; Jena [JEN] para acesso das Ontologias que descrevem os contextos de uso da aplicação; e OWL-S Mindswap [MIN] para construção dos Serviços Web Semânticos. O JADE foi adotado principalmente por estar em conformidade com os padrões estabelecidos pela *Foundation for Intelligents Physical Agents (FIPA)*, possuindo características como: serviço de nomes e páginas amarelas, transporte de mensagens, serviços de codificação e decodificação de mensagens e uma biblioteca de protocolos de interação. O Jena vem se

tornando um padrão para o desenvolvimento de Ontologias. O OWL-S Mindswap foi adotado por possuir uma documentação bastante completa.

Para facilitar o reuso do UBICK, no nível de projeto, os Agentes de Software foram representados em um modelo de componentes da UML [OMG04]. Dessa forma, o Engenheiro de software, não precisa ter conhecimentos sobre Agentes de Software para reutilizá-los.

Considerando a arquitetura, Cliente e Servidor, adotada para o UBICK, seu projeto foi organizado nos pacotes *UBICKClient* e *UBICKServer*. A Figura 4.12 apresenta o pacote *UBICKClient*, que é composto pelos seguintes componentes: *UserInterface*, *ContextAgentClient*, *OfflineRecords*, e *CalloutProtocolClient*.

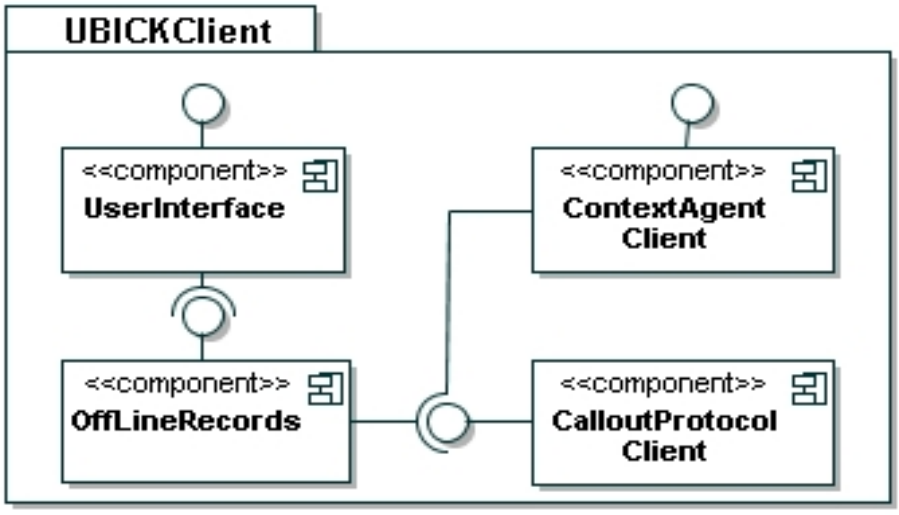

Figura 4.12. Pacote Client

O componente *UserInterface* é responsável pela apresentação das informações acessadas através do dispositivo móvel. O uso desse componente não é obrigatório, uma vez que um cliente pode não possuir interface (e.g., sensor, câmera). O componente *ContextAgentClient* monitora as informações relevantes ao contexto de uso da aplicação, enviando-as periodicamente ao

servidor. O componente *OfflineRecords* é usado para armazenar informações do cliente e do servidor, a fim de que estejam disponíveis em caso de falta de conectividade. As informações do cliente são temporariamente armazenadas num *Record Management System (RMS*). Posteriormente, essas são enviadas para o servidor usando a estratégia de sincronização de informações apresentada em [OGL07]. O componente *CalloutProtocolClient* é responsável pelo protocolo de comunicação entre os pacotes *Client* e *Server*.

A Figura 4.14 apresenta o pacote *UBICKServer*, que é composto pelos seguintes componentes: *ServerManager*, *AgentFactory*, *ServicesAgent*, *MobilityManager*, *AdaptationAgent*, *ContentTransferProtocol*, *Cache*, *ContextAgentSever*, *BrokerAgent*,*ContextRepository* e *CalloutProtocolServer*.

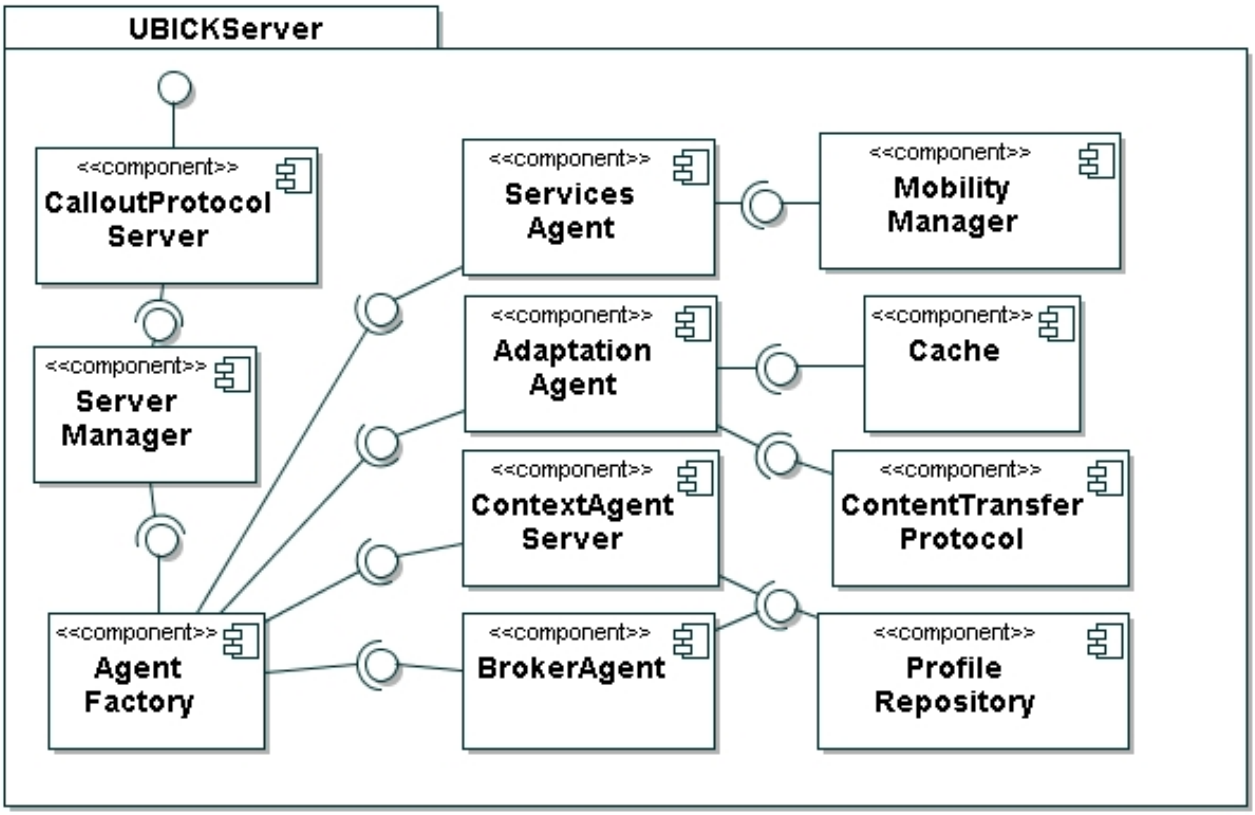

Figura 4.14. Pacote Server

O componente *ServerManagerAgent* gerencia o comportamento do servidor. Este recebe requisições do *CalloutProtocolServer* e, para cada requisição, instancia agentes através do componente *AgentFactory*, no padrão *Factory* [GAM95]. O componente *ServicesAgent* é um agente que busca, compõe e monitora a execução de Serviços Web Semânticos e que, de acordo com a política de mobilidade o componente *MobilityManager*, pode migrar para outros servidores. O componente *AdaptationAgent* é um agente que adapta conteúdos, acessados via *Client*, e interage com o componente *ServicesAgent* a fim de encontrar os serviços apropriados para uma adaptação particular. O componente *Cache* é utilizado para melhorar o desempenho do servidor. O componente *ContextAgentServer* é um agente que interage com o *ContextAgentClient* para atualizar informação contextual, a qual está armazenada no componente *ContextRepository*. O componente *BrokerAgent* é um agente que usa o padrão *Broker* [GAM95] e emprega Ontologias do *ContextRepository* para definir necessidades do usuário.

A Figura 4.15 apresenta o modelo de estados [OMG04] do componente *CalloutProtocolServer*, que recebe requisições provenientes do *UBICKClient* e intercepta requisições de conteúdo provenientes de navegadores. Nesse ultimo caso, esse componente opera de forma semelhante ao *proxy* apresentado em [SAN07]. O *CalloutProtocolServer* permanece no estado ***Aguardando*** até que uma requisição seja recebida (1) ou interceptada (2). No caso (1) esse componente identifica o *UBICKClient* e determina o tipo de requisição (e.g., atualização de contexto, acesso a serviço ou a conteúdo), e no caso (2) determina o contexto (e.g., navegador Web, dispositivo) via cabeçalho da requisição.

Figura 4.15. Modelo de Estados do *CalloutProtocolServer*

4.4. Implementação

Prosseguindo com o desenvolvimento do UBICK, fez-se a sua implementação, conforme as especificações do projeto. Assim, na linguagem alvo da implementação, foram implementados todos os seus componentes, como por exemplo, o responsável pela composição de serviços de adaptação. Na Listagem 4.1 é apresentado o algoritmo que deu origem ao código do componente para composição sucessiva.

Listagem 4.1. Algoritmo de Composição Seqüencial

```
List sequentialComposition (List composition, List allServices, Service newService, Format
finalFormat) {
  If(allServices == null) return null;
    Else{
      If(newService.outFormat == final){
        composition.add(newService);
      }Else{
        Int i = 0;
        While(allServices.get(i) != null){
         If(allServices.get(i).inFormat==
```

```
          composition.get(composition.length)){
            allServices.remove(allServices.get(i));
            composition.add(sequentialComposition (composition,
            allServices, allServices.get(i), final);
            i++;
          }
        }
      }
    }
  }
   return composition;
}
```

Esse algoritmo recursivo tem as seguintes entradas: uma composição de serviços (composition); uma lista de serviços que podem ser compostos(allServices); um serviço analisado na etapa atual da recursão (newService); e o formato para o qual o conteúdo deve ser adaptado (finalFormat). Na primeira chamada do método, essas entradas recebem os seguintes valores: composition=null, allSevices= lista de serviços disponíveis, newService= serviço candidato a participar da composição de serviços, e finalFormat= formato para o qual o conteúdo deve ser adaptado. A primeira verificação do algoritmo é uma condição de parada da recursão quando não houver serviços que possam ser compostos. Enquanto existem serviços, verifica-se se o serviço candidato pode realizar toda ou parte da adaptação do conteúdo, conforme o finalFormat. Caso positivo o serviço é removido da lista allServices e adicionado na lista de composições, e caso contrário, é descartado. A cada chamada recursiva, verifica-se a lista de serviços compostos já atende ao finalFormat. Caso positivo, encerra-se o algoritmo.

A Listagem 4.2 apresenta o algoritmo para composição paralela, que recebe como entrada todos os serviços disponíveis, os formatos de entrada e saída para cada conteúdo a ser adaptado. Este algoritmo pode utilizar o algoritmo seqüencial para compor serviços não disponíveis.

Listagem 4.2. Algoritmo de Composição Paralelo

```
List parallelComposition (List allServices, List outputs, List inputs) {
  List composition = new List();
  for(int i = 0; i < outputs.length(); i++){
      for(int j = 0; j < allServices.length(); j++){
         if(allServices[j].input == inputs[i]
           && allServices[j].output == outputs[i]){
              composition.add(allServices[j]);
         }else{
              composition.add(sequentialComposition(null,
              allServices,null,outputs[i]));
         }
    }
    return composition;
}
```

4.5. Testes

Como apresentado em [SAN07], para avaliar uso de Serviços Web Semânticos, foi definida uma configuração de rede envolvendo 04 computadores Pentium 4-2.5 Ghz Dual Core com 2 GB de memória RAM, todos usando Windows XP. Num computador foi instalado um servidor, com uma base de dados contendo os acessos ao proxy do laboratório de graduação do Departamento de Computação (DC) da UFSCar, enquanto nos demais foram instalados Serviços Web Semânticos para adaptação de páginas Web. Os testes possibilitaram calcular o tempo de resposta para cada requisição, em diferentes configurações do Server, enfatizando a análise de desempenho de um servidor para adaptaçãoes de páginas Web, desenvolvido a partir do UBICK, quando submetido a um grande número de acessos.

Durante 09 horas foram coletadas milhares de requisições ao proxy. Somente as 150 primeiras estão ilustradas nos gráficos das Figuras 4.16 e 4.17, já que o comportamento presente nesse conjunto de requisições repete-se ao longo do tempo. Conseqüentemente, em cada gráfico, os padrões das curvas dos Serviços Web Semânticos e dos Serviços Web tradicionais são os

mesmos. Em cada uma dessas figuras estão representados os testes realizados com o uso de Serviços Web tradicionais e com o uso de Serviços Web Semânticos, este último via *API OWL-S Mindswap*. Os testes visualizados na Figura 4.16 são com o cache do serviço desativado enquanto que na Figura 3.16 são com o cache do serviço ativado.

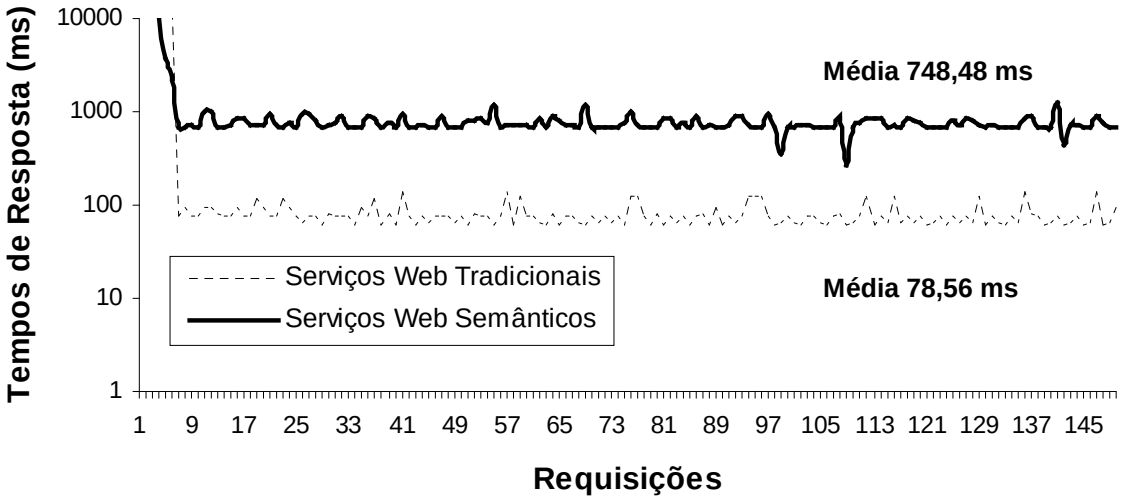

Figura 4.16. Tempos de resposta às requisições com o cache desativado

De acordo com a Figura 4.17, o uso de Serviços Web Semânticos torna o tempo de resposta às requisições em média 10 vezes maior que o tempo de resposta com o uso de Serviços Web tradicionais. Esse resultado já era esperado porque o emprego de Serviços Web Semânticos aumenta a complexidade do processo de adaptação de conteúdo como um todo e porque a API usada possui um baixo desempenho, já que foi desenvolvida para fins acadêmicos. Entretanto, essa perda quantitativa no tempo de resposta é largamente compensada pelo ganho qualitativo propiciado por essa complexidade adicional, ou seja: a capacidade de realizar inferências sobre o contexto de entrega para determinar os serviços necessários à execução de uma determinada adaptação; a busca automática desses serviços; a composição também automática desses serviços.

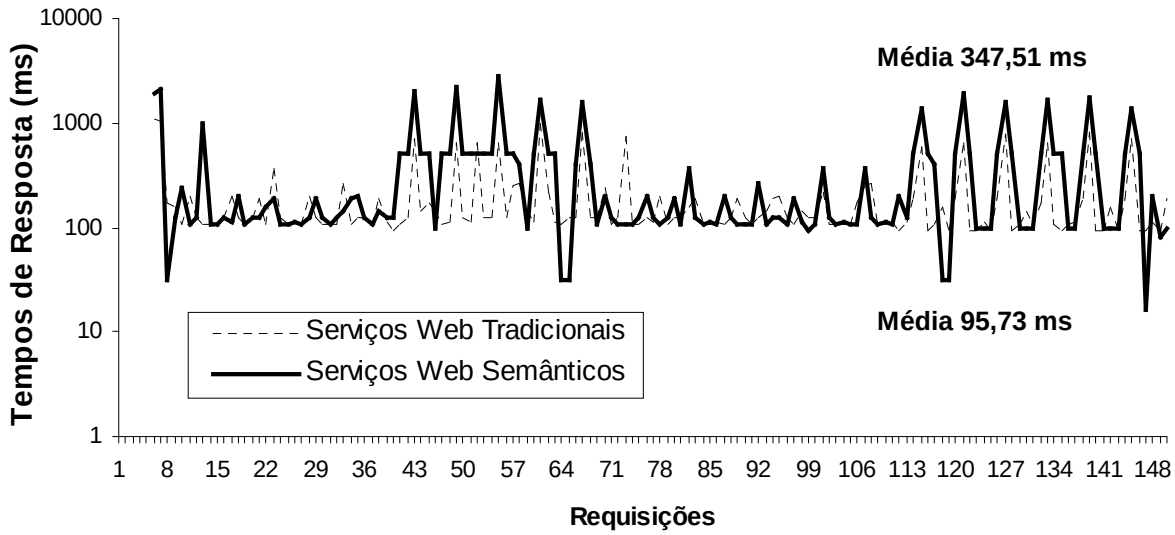

Figura 4.17. Tempos de resposta às requisições com o cache ativado

De acordo com a Figura 4.17, o uso do cache acarretou num pequeno aumento no tempo de resposta, quando Serviços Web tradicionais são empregados, e numa redução considerável desse tempo, quando os Serviços Web Semânticos são utilizados. Nesse sentido, foi usada a mesma escala logarítmica nos gráficos 4.16 e 4.17, o que aproxima as duas curvas em 4.17, para ressaltar que o simples emprego de cache diminui significativamente a perda de desempenho causada pelo uso de Ontologias e Serviços Web Semânticos. O aumento é devido ao *overhead*, introduzido pelo acesso ao disco durante a inserção e a recuperação de dados no cache, e a redução é devida à presença de conteúdos já adaptados no cache, evitando assim a repetição de todo o processo de adaptação. Outro aspecto a ser ressaltado é a presença de dois conjuntos de picos de tempos de resposta, que estão nas faixas 43 a 64 e 115 a 140. Esses picos são decorrentes do acesso a um conjunto de páginas ainda não presentes no cache. Já que é comum o uso de

cache em *Servidores*, esses resultados justificam ainda mais o emprego de Serviços Web Semânticos para a adaptação de conteúdo na Internet.

Para avaliar o uso de Agentes de Software, como apresentado em [SAN07a], foi configurada uma rede com 05 computadores Pentium 4- 3 Ghz com 1 GB de memória RAM, e Sistema Operacional Windows XP. Três computadores foram utilizados como Servidores de Adaptação (um local e dois remotos), um como Servidor Dedicado de Adaptação, e um simulando um dispositivo do usuário. A Figura 4.18 mostra o resultado dos testes, que consistiram do cálculo do tempo de resposta para cada requisição, em diferentes configurações dos Servidores. Na primeira configuração as adaptações foram sem utilizar agentes, e as requisições foram colocadas numa fila. Na segunda configuração, para cada requisição um novo Agente Gerente foi criado.

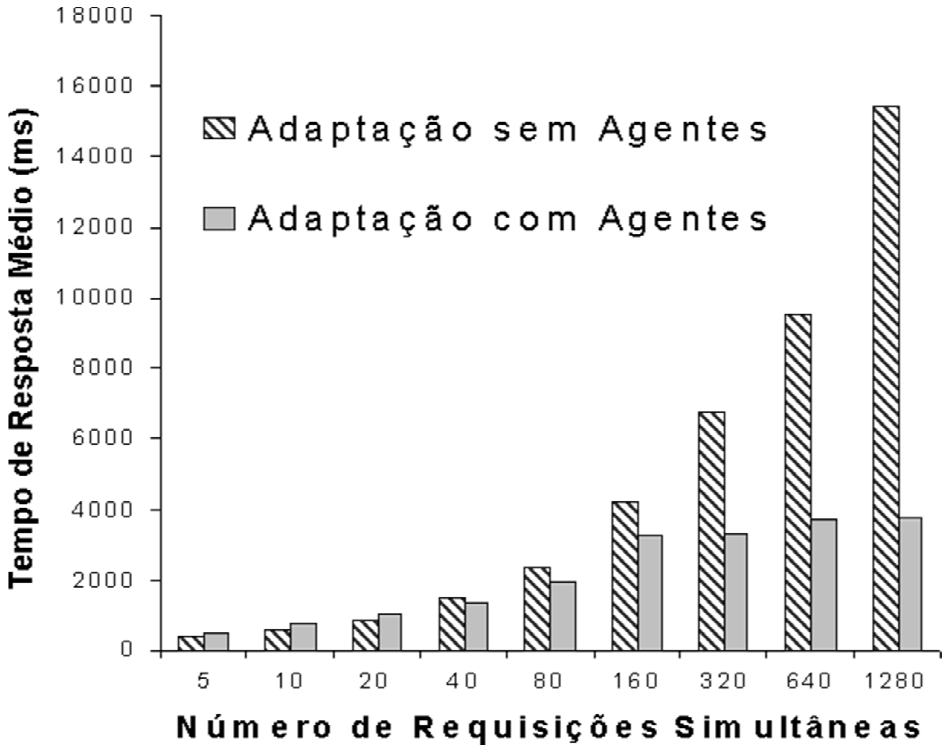

Figura 4.18. Tempo de Resposta Médio em Relação à quantidade de acessos simultâneos

Analisando os tempos de resposta da adaptação sem agentes verificou-se um crescimento exponencial do tempo de resposta em relação ao número de requisições simultâneas. Na adaptação com agentes o tempo médio de resposta tem crescimento linear a partir de 160 requisições simultâneas. Dessa forma, pelos resultados obtidos, pode-se concluir que os agentes, além de ampliar a capacidade de adaptação, melhoram o desempenho quando o número de requisições simultâneas é grande.

4.6. Conclusões

Este capítulo apresentou o desenvolvimento do UBICK, construído para apoiar a abordagem AdeSCoU. No desenvolvimento do UBICK foram empregadas Ontologias, Serviços Web Semânticos e Agentes de Software. Ontologias foram usadas para descrição de contexto. Serviços Web Semânticos foram usados para implementação de serviços das aplicações ubíquas, como os serviços para adaptação de conteúdo. Finalmente, Agentes de Software foram usados para gerenciar as tarefas das aplicações ubíquas.

Capítulo 5

Abordagem AdeSCoU

Combinando as idéias e conceitos da Computação Ubíqua, de Ontologias, dos Serviços Web Semânticos, de Agentes de Software, e de componentes de software, com as do *RUP (Rational Unified Process)* [KRU03], este projeto investigou e desenvolveu uma abordagem para orientar o desenvolvimento de aplicações ubíquas.

O RUP é um *framework* para processos [KRU03] que define uma grande quantidade de elementos (papéis, artefatos, fluxos de trabalhos) e boas práticas de Engenharia de Software. Como framework, o RUP possibilita sua instanciação conforme as necessidades específicas de um determinado processo de desenvolvimento de software. Assim, dependendo dos elementos utilizados do RUP tem-se uma instância que constitui um processo de desenvolvimento completo para um projeto ou um conjunto de projetos. Como exemplos, têm-se as instâncias do RUP para SOA [IBM] e para Governança de software [IBMa]. Nesse sentido, a *Abordagem para o Desenvolvimento de Software na Computação Ubíqua (AdeSCoU)*, proposta neste projeto de pesquisa, é uma instância do RUP orientado para a Computação Ubíqua. As principais customizações do RUP na AdeSCoU relacionam-se com o emprego Ontologias, Serviços Web Semânticos e Agentes de Software. Como no RUP, o processo de desenvolvimento proposto é dividido em fases e disciplinas

ortogonais entre si. A Figura 5.1 ilustra o processo de desenvolvimento que evolui com as fases, nas quais são realizadas todas as disciplinas, com maior ou menor ênfase. Semelhante ao RUP, ao longo do tempo, a abordagem AdeSCou evolui com as fases: Concepção, Elaboração, Construção e Transição. Durante estas fases, a aplicação que está sendo desenvolvida evolui dos requisitos em níveis mais abstratos até uma aplicação implementada e disponível para execução. Outras abordagens, genéricas como as apresentadas em [SAN07b, SAN07c] foram investigadas para orientar o desenvolvimento da abordagem proposta.

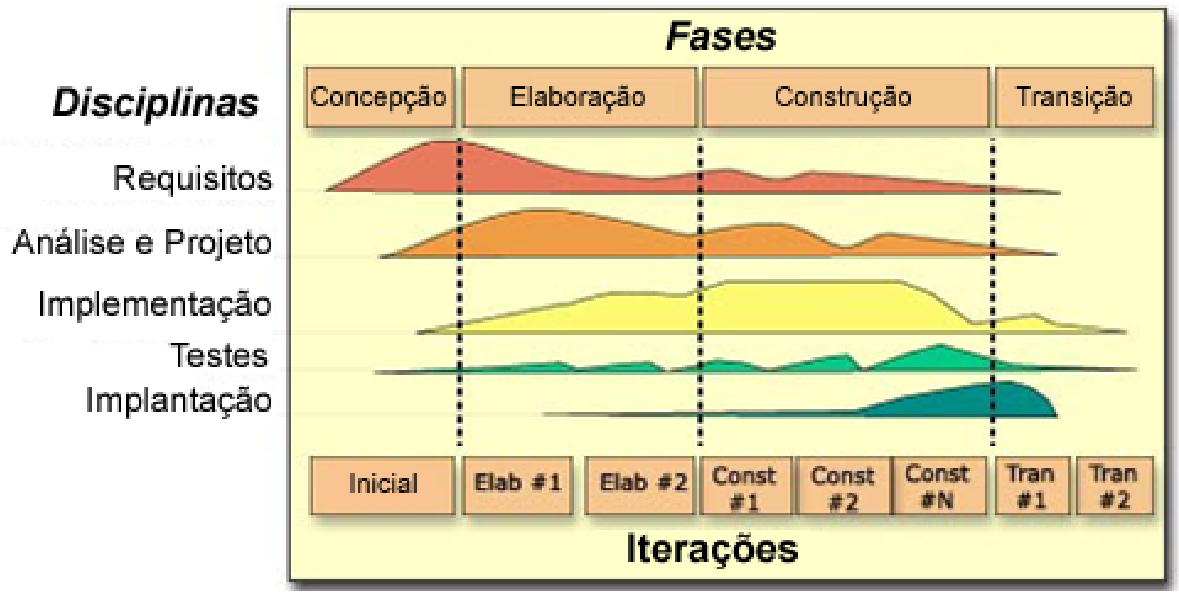

Figura 5.1. Fases e Disciplinas da Abordagem [KRU03]

Na AdeSCoU, as seguintes disciplinas são consideradas: Requisitos, Análise e Projeto, Implementação, Testes e Implantação. As atividades realizadas em cada disciplina e suas intensidades em cada fase, dependem do projeto. É importante salientar que as disciplinas Engenharia de Negócios, Configuração e Controle de Modificação, Gerência de Projeto e Ambiente, embora presentes no RUP, não são tratadas na abordagem, devido a limitações de tempo para realização dessa pesquisa.

Na seqüência apresentam-se os principais conteúdos das disciplinas utilizadas na abordagem AdeSCoU para o desenvolvimento de aplicações ubíquas.

5.1. Disciplinas

As disciplinas utilizadas pela abordagem AdeSCoU serão apresentadas através dos seus fluxos de trabalho, conforme o RUP. Na apresentação dessas disciplinas utilizadas do RUP, as contribuições da abordagem AdeSCoU, para atender o paradigma da Computação Ubíqua, são destacadas com o estereotipo <<**ubicomp**>>.

5.1.1. Requisitos

Os principais objetivos dessa disciplina no desenvolvimento da aplicação são: identificar seus requisitos; definir seus limites, e estabelecer um contrato entre os Engenheiros de Software e os Stakeholders[5]. Esta disciplina visa obter uma compreensão completa dos requisitos, o que é fundamental para o planejamento, o desenvolvimento e a aceitação dos resultados do projeto.

A Figura 5.2 apresenta o fluxo de trabalho principal da disciplina Requisitos da AdeSCoU. Partindo de um determinado problema, verifica-se a maturidade do projeto (novo ou já existente). Em novos projetos, o problema é analisado, como no RUP, para identificar os Stakeholders envolvidos e definir os limites da aplicação.

Prosseguindo, faz-se a elicitação das necessidades dos Stakeholders, que interagem com a aplicação. Como no RUP, essa atividade visa compreender as necessidades dos envolvidos com o projeto, através da identificação de um vocabulário comum, e do desenvolvimento da Visão do

[5] *Um Stakeholder é qualquer pessoa ou organização que possui interesse em relação aos resultados do projeto.*

software. A Visão é representada por um artefato que contém a descrição dos requisitos centrais do projeto, formando a base contratual para requisitos técnicos mais detalhados. Na realização dessas atividades podem ser utilizadas técnicas como StoryBoards [KHA06], Mind Maps, Entrevistas e Brainstorms [PRE02]. Enquanto o problema não estiver bem definido repetem-se essas atividades refinando-se os seus artefatos.

Com o problema corretamente definido, prossegue-se na atividade **Definir a Aplicação** onde são refinadas as especificações dos requisitos. No caso de uma aplicação ubíqua devem ser também definidos os requisitos relacionados com as suas adaptações de comportamentos e conteúdos, denominados Requisitos Adaptáveis. Baseado nos trabalhos de [SIT07, BER05], que vem investigando a identificação, modelagem, análise, e elicitação de requisitos na Computação Ubíqua, foi adotada na abordagem AdeSCoU a utilização de Cenários [LAM01], que facilitam o desenvolvimento de Casos de Uso, de uma forma simples e flexível. Essa técnica melhora o processo de especificação de requisitos através de um maior envolvimento dos Stakeholders. Outra idéia adotada com base nesses trabalhos relaciona-se com a identificação dos requisitos adaptáveis conforme o contexto da aplicação [SIT07] (e.g., dispositivo de acesso, localização, rede de acesso). A aplicação deve atender esses requisitos com modificações de contexto que são disparadas por condições internas dependendo, por exemplo, da localização, da rede de acesso e das características do dispositivo.

Prosseguindo, define-se o escopo da aplicação, a fim de adequar seus requisitos aos recursos disponíveis (tempo, pessoas e dinheiro).

As atividades dessa disciplina são repetidas até que os requisitos estejam claramente definidos.

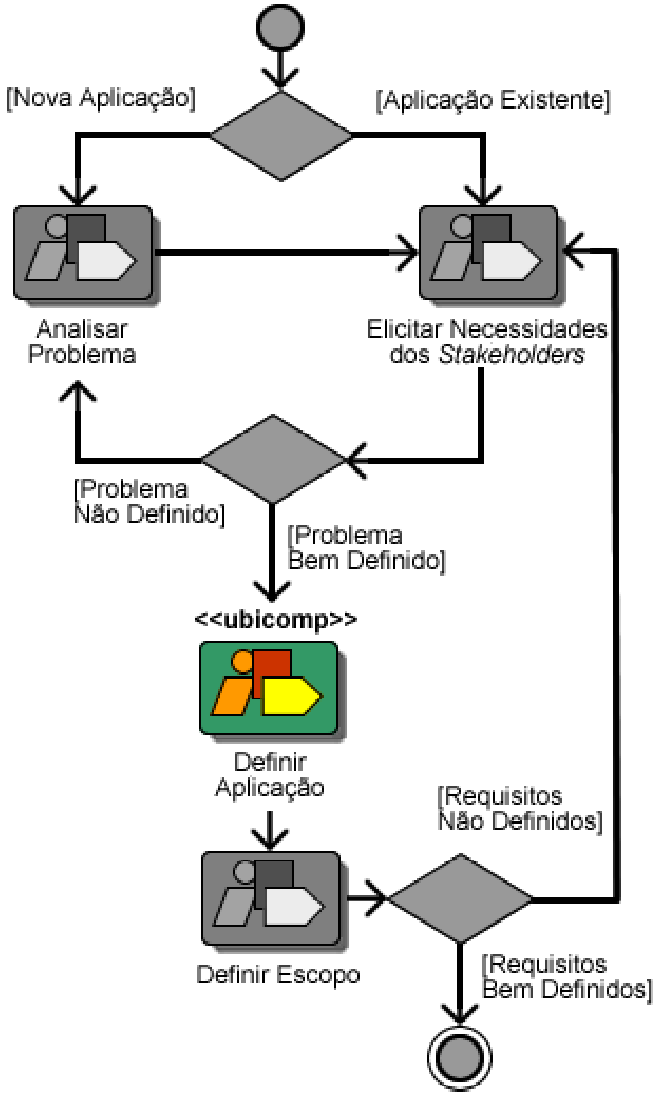

Figura 5.2. Fluxo de Trabalho da disciplina Requisitos

5.1.2. Análise

A disciplina Análise tem como objetivo transformar os requisitos, com técnicas da UML, em modelos independentes de plataforma. Nessa disciplina, são analisados os comportamentos da aplicação, a partir dos requisitos representados em mind-maps, storyboards, casos de uso, e outra técnicas utilizadas na disciplina Requisitos. Essas especificações são analisadas e transformadas em um conjunto de métodos e atributos de classes. Outra

atividade dessa disciplina consiste em refinar os requisitos adaptáveis representando-os em modelos de ontologias.

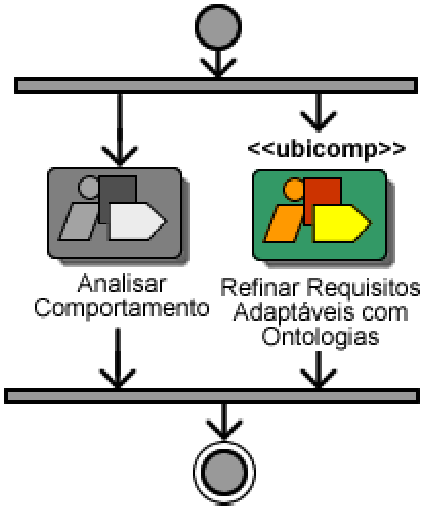

Figura 5.3. Fluxo de trabalho principal da disciplina Análise

5.1.3. Projeto

A disciplina de Projeto transforma os modelos de análise em modelos dependentes de plataforma. A Figura 5.4 apresenta o fluxo de trabalho principal para a disciplina Projeto. Em Definir Plataforma, são tomadas decisões sobre o sistema operacional, sistema gerenciador de banco de dados, servidores de aplicação e web, linguagem de programação, protocolos, modelos de dispositivos e outros padrões de desenvolvimento escolhidos.

Na AdeSCoU, a Arquitetura da aplicação é organizada em Clientes e Servidores, baseando-se na arquitetura do UBICK. Para definir a arquitetura da aplicação parte-se do modelo de classes da Análise, refinado conforme as decisões de projeto. Esses modelos de classes de projeto são analisados e servem de base para projetar os componentes da aplicação.

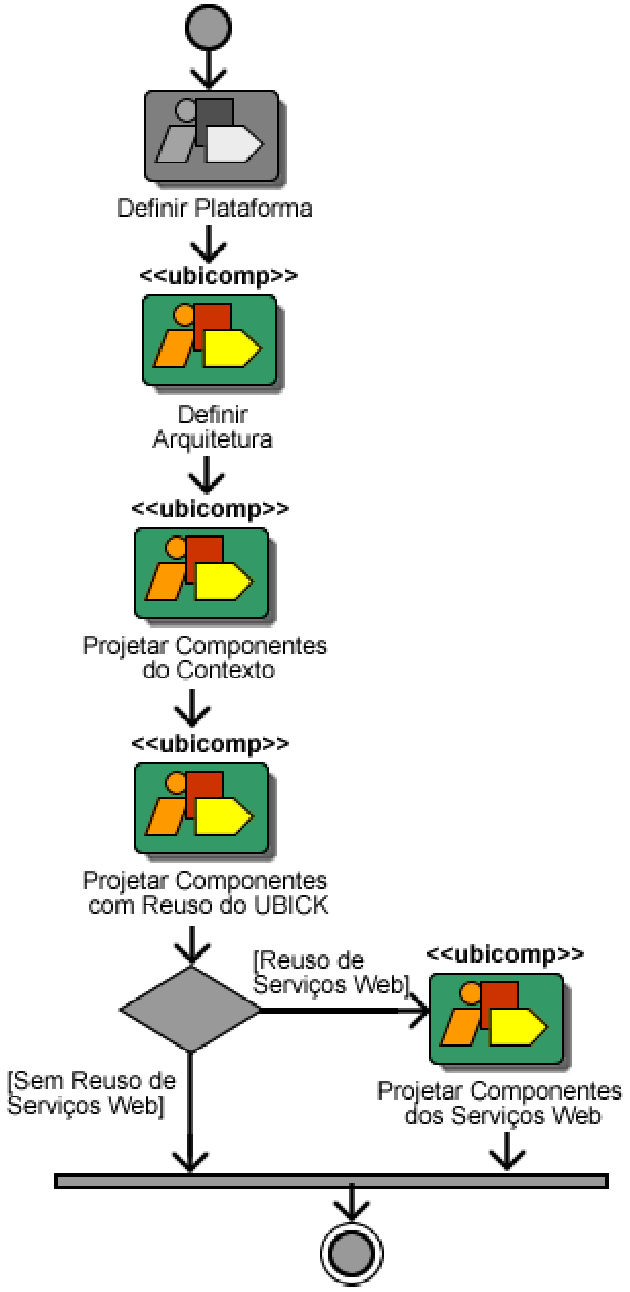

Figura 5.4. Fluxo de trabalho principal da disciplina Projeto

Prosseguindo, faz-se o projeto dos componentes com base nos modelos de classes. Para melhor compreensão, a atividade projetar componentes do RUP, foi decomposta nas atividades Projetar Componentes do Contexto e Projetar demais Componentes, ambas com Reuso do UBICK, e opcionalmente Projetar Componentes dos Serviços Web. Em Projetar Componentes do Contexto com reuso do UBICK, faz-se o refinamento das Ontologias dos

requisitos adaptáveis integrando-as com a Ontologia do UBICK. Em seguida, faz-se o projeto dos demais componentes que fazem reuso do UBICK, integrando-os com os componentes do contexto. Baseado no conhecimento dos recursos oferecidos pelo UBICK o Arquiteto da Aplicação decide sobre as possíveis colaborações do framework, que realizam os diferentes comportamentos da aplicação. Para estruturar a aplicação na arquitetura cliente e servidor, os componentes são organizados conforme os pacotes Client e Server do UBICK.

Opcionalmente, o desenvolvedor projeta os componentes dos Serviços Web[6]. Para decidir quais componentes do Servidor serão projetados como Serviços Web, usam-se os critérios: da (a) disponibilidade de Serviços Web na Internet, com as funcionalidades requeridas pelos componentes a serem projetados; (b) necessidade de distribuição das tarefas dos componentes (e.g., para melhorar o desempenho, a confiabilidade, para que esteja disponível para vários servidores); e (c) freqüência de modificação desses componentes, uma vez que os Serviços Web podem ser trocados em tempo de execução.

As especificações são refinadas até obter um projeto integrado dos componentes específicos da aplicação, com os reutilizados do UBICK.

5.1.4. Implementação

Nessa disciplina organiza-se o código da aplicação e implementam-se os componentes do projeto (e.g., código-fonte, binários, executáveis, serviços). Na AdeSCoU faz-se a implementação dos componentes baseado na reutilização de componentes já disponíveis. O programador codifica, compila e realiza os

[6] *Nesse contexto, Serviços Web podem ser vistos como componentes cujas interfaces são disponíveis através de mensagens baseadas em eXtensible Markup Language (XML) [YIJ07].*

testes unitários (e.g., usando o JUnit[7] [JUN]) dos componentes conforme o projeto. Nesta disciplina utiliza-se o Teste Estrutural para avaliar o comportamento interno do componente de software. Essa técnica trabalha diretamente sobre o código-fonte do componente de software para avaliar aspectos tais como: teste de condição, teste de fluxo de dados, teste de ciclos e teste de caminhos lógicos.

A Figura 5.5 apresenta o fluxo de trabalho da disciplina Implementação, que compreende as atividades de Implementar Componentes Específicos da Aplicação com reuso do UBICK, e opcionalmente Implementar Componentes dos Serviços WEB.

Os Componentes dos Serviços Web podem já ter suas implementações providas e disponíveis para reuso. Portanto, inicialmente, faz-se uma busca desses serviços, visando reduzir o tempo e custos da implementação. Caso o serviço necessário não tenha sido encontrado, o programador deve prover a sua implementação, conforme as especificações do projeto. Assim, como os serviços já existentes, o novo serviço implementado fica disponível para o reuso. Para serem reutilizados, os componentes devem ser descritos semanticamente em OWL-S. Uma vantagem dos componentes serviços web, é que seu código pode ser substituído em tempo de execução, evitando interrupções na execução dos servidores.

[7] *O JUnit é um framework open-source, que facilita a criação de testes unitários, uma vez que possibilita a criação de programas que realizem os testes pelo programador*

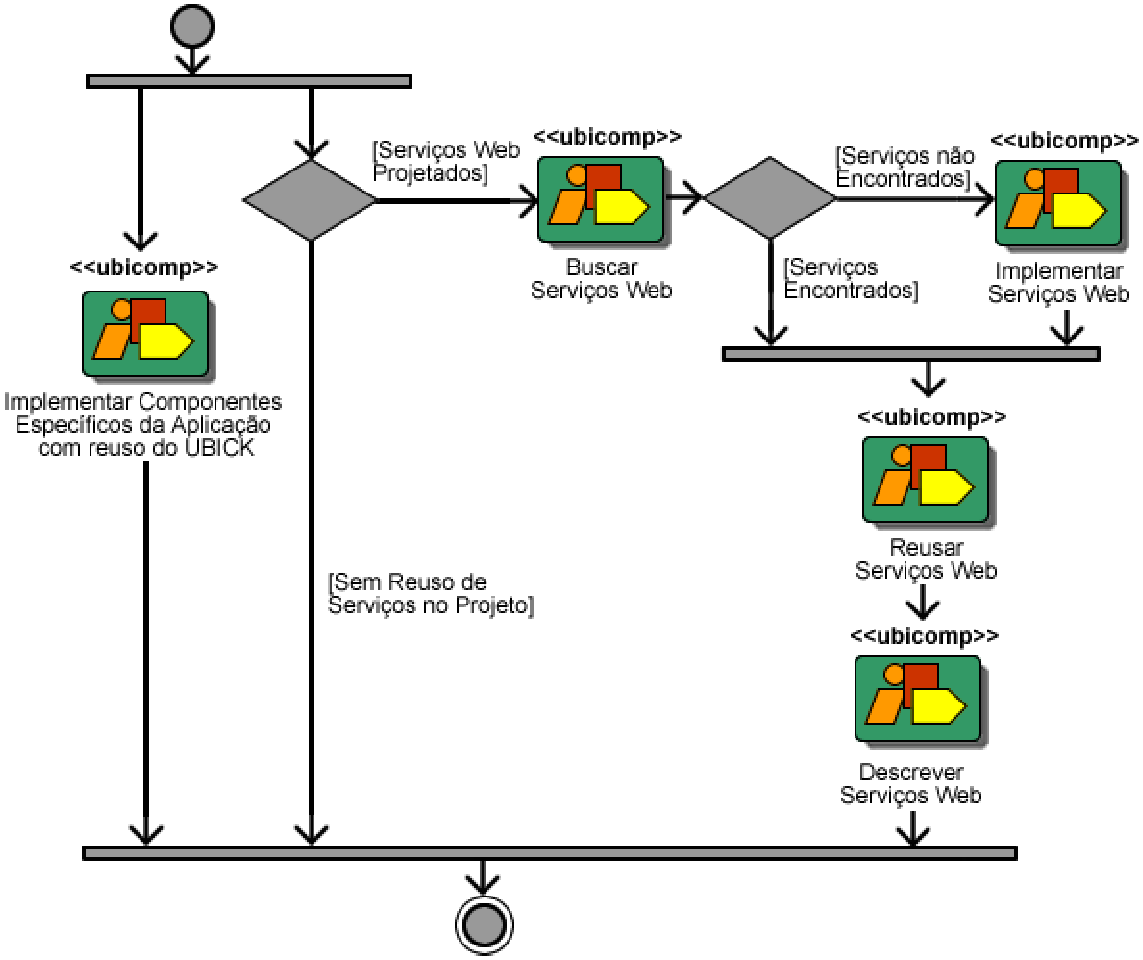

Figura 5.5. Fluxo de trabalho principal para disciplina Implementação

5.1.5. Testes

Seguindo o RUP, os objetivos da disciplina Testes são: encontrar falhas na aplicação, aconselhar a gerencia sobre a qualidade da aplicação e validar as aplicações desenvolvidas em relação aos requisitos especificados. Na abordagem AdeSCoU os testes devem verificar se os requisitos da aplicação foram atendidos, considerando os aspectos de distribuição, apresentação do conteúdo adaptado, e ciência de contexto. Considerando que os testes unitários dos componentes foram realizados junto com suas implementações, nessa disciplina faz-se o teste integrado desses componentes estruturados em Clientes e Servidores.

A Figura 5.6 apresenta o fluxo principal de trabalho da disciplina Testes. Conforme ilustra o fluxo, ao realizar o teste da aplicação verifica-se ou não a ocorrência de erros. No caso de ser encontrado algum erro este é analisado para definir o fluxo de retorno para sua correção. Dependendo do erro, pode-se retornar a qualquer uma das disciplinas, Requisitos, Análise, Projeto ou Implementação, para corrigi-lo.

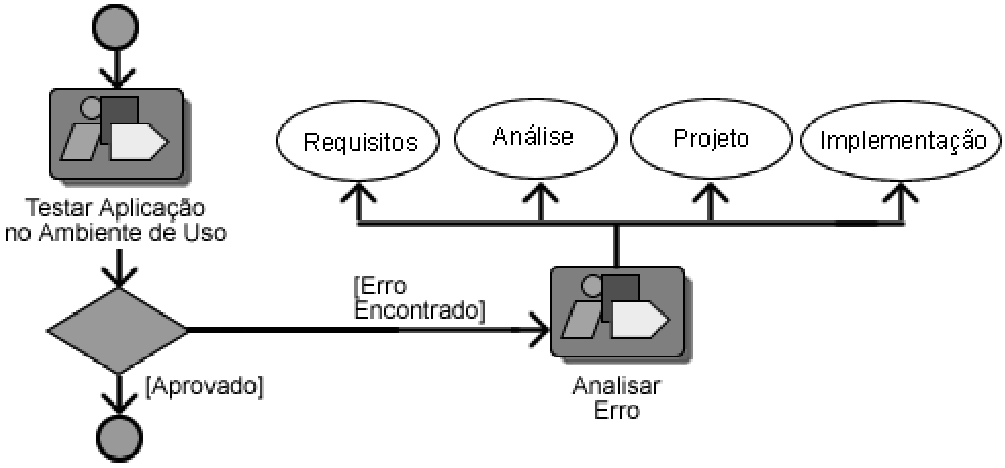

Figura 5.6. Fluxo de trabalho principal para disciplina Testes

Nesta disciplina, são realizados Testes Funcionais, em que o componente de software a ser testado é acessado pela sua interface como se fosse uma caixa-preta, ou seja, não é considerado o seu comportamento interno. Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido.

5.1.6. Implantação

Nessa disciplina o Gerente da Implantação faz o empacotamento da aplicação para a sua distribuição, torna-a disponível para execução e faz o treinamento dos seus usuários.

A Figura 4.6 apresenta o fluxo de trabalho principal dessa disciplina, que tem início com o empacotamento da versão da aplicação a ser implantada. A

versão produzida é organizada nos pacotes: cliente e servidor. No pacote servidor têm-se os Serviços Web que serão distribuídos conforme a arquitetura da rede onde a aplicação será executada. Em seguida, a aplicação é disponibilizada para execução. Com a aplicação disponível, seus usuários devem ser treinados. O treinamento inclui instruções para instalar, usar e manter a aplicação desenvolvida.

Figura 5.7. Fluxo de trabalho principal para disciplina Implantação

As disciplinas apresentadas são realizadas em cada fase da abordagem conforme apresentado a seguir.

5.2. Fases da Abordagem proposta

A abordagem proposta utiliza os elementos apresentados (papéis, atividades, artefatos, fluxos de trabalho e disciplinas), em 4 fases básicas (Concepção, Elaboração, Construção e Transição). Cada fase evolui em várias iterações, e

cada iteração representa um ciclo completo (Requisitos, Análise, Projeto, Implementação, Testes e Implantação). Cada fase possui um conjunto de objetivos, atingidos através de atividades essenciais, e são finalizadas quando se verifica que um marco, denominado *milestone* [KRU03], foi alcançado.

5.2.1. Concepção

Os objetivos da fase de Concepção são definir os riscos e o escopo do projeto. Tais objetivos são alcançados através das atividades essenciais, realizadas com maior ou menor ênfase nas disciplinas apresentadas na seção 4.1.

Nessa fase as atividades essenciais são: a) *Analisar o Problema*, *Elicitar Necessidades dos Stakeholders* e *Definir o Escopo*, da disciplina Requisitos; b) *Definir uma Arquitetura Abstrata*, da disciplina Análise. Estas atividades irão facilitar a definição da viabilidade do projeto.

A *milestone* dessa etapa é chamada de *Life Cycle Objective* (*LCO*), além das previstas pelo RUP, é importante que ao final dessa etapa os principais cenários de utilização da aplicação que será desenvolvida estejam identificados. Enquanto essa *milestone* não for atingida o projeto consideravelmente reformulado.

5.2.2. Elaboração

O objetivo da fase de Elaboração é a análise da aplicação, para estabelecimento de uma arquitetura e de um plano de desenvolvimento. Como citado em [KRU03], nessa fase é obtida uma visão geral da aplicação (“one mile wide, one inch deep vision”) e é criada de uma linha base (baseline) a partir do refinamento da fase anterior. Dessa forma, tem-se uma versão do projeto que caracterize um ponto de referência para o restante do desenvolvimento.

Nessa fase as atividades essenciais são: a) *Analisar Problema*, da disciplina Requisitos; b) *Analisar Comportamento* e *Refinar Requisitos Adaptáveis com Ontologias*, da Análise; e, c) *Definir Plataforma* e *Refinar a Arquitetura*, da disciplina Projeto. A partir da Arquitetura refinada, estará disponível uma linha base para o projeto.

A milestone dessa etapa é chamada de Life Clycle Architeture (LCA). Nesse momento, devem ser examinados os objetivos e o escopo do projeto, a opção de arquitetura e os riscos devem estar resolvidos.

5.2.3. Construção

Na fase de construção os componentes são desenvolvidos e integrados, e todas as características são testadas. Essa fase representa uma transição entre o desenvolvimento abstrato das fases anteriores, para o desenvolvimento de um produto para a fase de transição.

Nessa fase as atividades essenciais são: a) *Implementar Componentes Específicos da Aplicação com reuso do UBICK* e *Projetar Componentes dos Serviços Web*, de Projeto; c) *Implementar Componentes Específicos da Aplicação com reuso do UBICK*, *Buscar Serviços Web*, *Implementar Serviços Web*, *Reusar Serviços Web* e *Descrever Serviços Web*; e d) *Testar Aplicação no Ambiente de Uso*.

A fase de construção é finalizada, no *milestone Initial Operational Capability* (*IOC*), quando a aplicação está pronta para uso, sem expor o projeto a grandes riscos. Além do software, um manual do usuário e a documentação da versão atual devem estar prontos.

5.2.4. Transição

O propósito da fase de transição é entregar o software aos usuários finais. Durante essa fase, novas questões podem surgir requerendo correção de problemas, novas versões, ou finalização de componentes que foram adiados.

Nessa fase as atividades essenciais são: *Empacotar Aplicação, Disponibilizar para Uso* e *Treinar Usuários*, da disciplina Transição. Esta última fase inclui: entregar, treinar, apoiar e manter o produto até que os usuários estejam satisfeitos, concluindo a *milestone Product Release* (*PR*), que encerra a fase e o ciclo de desenvolvimento.

5.3. Conclusões

Esse capítulo apresentou uma abordagem que sistematiza a utilização dos resultados da pesquisa apresentada nos capítulos anteriores. Essa abordagem é em grande parte facilitada pela reutilização do framework UBICK e sua Ontologia, e pelos Serviços Web.

Essa abordagem consiste de uma instância do RUP, que evolui através das fases Concepção, Elaboração, Construção e Transição. Em todas as fases ocorrem as seguintes disciplinas: Requisitos, Análise e Projeto, Implementação, Testes e Implantação. Cada uma dessas disciplinas possui um conjunto de atividade, papéis, artefatos e um fluxo de trabalho principal. A abordagem proposta modifica alguns desses elementos a fim de facilitar o desenvolvimento de software para Computação Ubíqua.

Caso ocorra algum problema numa das atividades, o ciclo de desenvolvimento pode retornar em qualquer atividade das disciplinas anteriores, a fim de realizar os ajustes necessários.

Capítulo 6

Estudo de Caso

Este capítulo apresenta o uso da AdeSCoU no desenvolvimento de uma aplicação denominada Portfólio Reflexivo Eletrônico (PRE) [PER07, SAN08, MAR08], para apoiar o processo de ensino e aprendizagem no curso de Medicina da Universidade Federal de São Carlos (UFSCar).

O curso de Medicina da UFSCAR foi criado em 2005 com uma proposta inovadora tanto na organização curricular quanto nas atividades educacionais. Os estudantes, ao longo do curso, atuam em cenários diversificados, tais como domicílios, creches, escolas, instituições para idosos e pessoas com necessidades especiais, Unidades de Saúde da Família (USFs), ambulatórios de especialidades e serviços hospitalares, no sentido de desenvolverem capacidades para o cuidado integral à saúde das pessoas e da comunidade.

A organização curricular é orientada ao desenvolvimento de competências e estruturada em Unidades Educacionais, que por sua vez são constituídas de atividades (Situação Problema, Estações de Simulação, Prática Profissional). Essas unidades substituem a estrutura segundo disciplinas, uma vez que os pressupostos curriculares orientam-se por uma abordagem interdisciplinar. Na proposta pedagógica desse curso [CON05] o processo de ensino-aprendizagem utiliza princípios da Aprendizagem Baseada em Problemas (ABP) [COO06, BLI95].

O Portfólio Reflexivo (PR) é um instrumento de registro e reflexão, realizados de forma sistemática sobre a trajetória e as práticas desenvolvidas pelos estudantes nas Unidades Educacionais do Curso de Medicina da UFSCar. A utilização de portfólios tradicionais, cuja mídia empregada é o papel, possui desvantagens em relação à manutenção (e.g., perda ou esquecimento de itens do portfólio necessários numa certa atividade), ao acesso (e.g., falta de suporte para recuperação de informações específicas), ao transporte (e.g., um portfólio pode atingir um grande volume) e à própria utilização do mesmo (e.g., em atividades práticas) [NIG05].

Na tentativa de automatizar o Portfólio Reflexivo foi desenvolvida uma primeira versão, denominada Portfólio Reflexivo Eletrônico, seguindo os passos da abordagem AdeSCoU. Os principais recursos para o desenvolvimento dessa aplicação foram os laboratórios do Departamento de Computação (DC) e os membros do Grupo de Computação Ubíqua (GCU), da UFSCar.

6.1. Requisitos

Inicialmente investigou-se o problema, pesquisando trabalhos, com características semelhantes ao PRE, para conhecer melhor o domínio do problema e definir os seus limites. Prosseguindo, foram identificados os docentes e estudantes do curso de Medicina da UFSCar como principais stakeholders responsáveis pelo processo de ensino e aprendizagem. Sessões de *brainstorms* e entrevistas foram realizadas para elicitação das necessidades desses stakeholders, inicialmente, especificadas através da Visão e do Vocabulário da aplicação.

A Visão para o PRE é a seguinte: Trata-se de uma aplicação educacional, com múltiplos cenários (reais e simulados) com diferentes atores

(e.g., estudantes e docentes), muitas vezes geograficamente distribuídos (e.g., comunidades, domicílios, ambulatórios, hospitais, salas de pequenos grupos, anfiteatros, salas de professores e laboratórios), que necessitam recuperar, manipular, trocar e armazenar informações para a tomada de decisão, usando diversos tipos de dispositivos móveis (e.g., celulares, smartphones, pagers, tablets, PDAs) e fixos (e.g., desktops, laptops), caracteriza um ambiente tipicamente ubíquo.

Os principais termos do Vocabulário da Situação Problema são:

a) Docente, Estudante e Preceptor: representam respectivamente, professores do Departamento de Medicina da UFSCar, alunos do curso de Medicina da UFSCar, e profissionais da rede municipal de saúde (e.g., médicos, enfermeiras) que auxiliam os estudantes durantes suas atividades práticas;

b) Problema: uma situação ou texto, cujo papel é disparar uma discussão a fim de que um novo conhecimento seja formado;

c) Pequeno Grupo: grupo formado por um ou dois docentes, e no máximo oito estudantes, que atua durante um semestre, discutindo soluções para vários Problemas;

d) Facilitador e Co-Facilitador: são papéis dos Docentes. Facilitador tem responsabilidade de apresentar os Problemas e guiar os Pequenos Grupos, a fim de auxiliar a criação de conhecimento. Co-Facilitador, geralmente, é representa um docente que está sendo treinado no processo de ensino e aprendizagem;

e) Síntese Provisória, Nova Síntese e Questões de Aprendizagem: o primeiro representa o conhecimento prévio de um Estudante, em relação a um determinado Problema; o segundo representa questões levantadas por um

Pequeno Grupo durante uma reunião; e o terceiro contém o resultado consolidado dos estudos de um Estudante sobre um determinado Problema.

Em seguida, foram identificados os Atores que interagem com as aplicações do domínio do problema e seus Casos de Uso, conforme ilustra a Figura 6.1. Os atores do PR são: Docente, especializados em Facilitador e Co-Facilitador; Estudante e Preceptor.

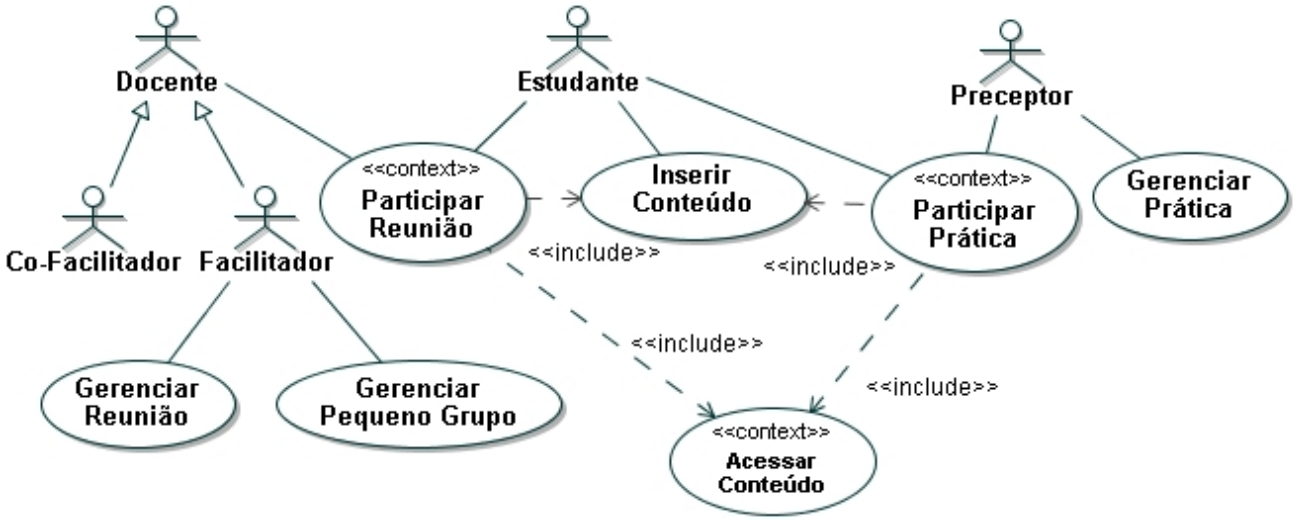

Figura 6.1. Principais Casos de Uso do PRE

Em Inserir Conteúdo, os estudantes inserem os conteúdos relacionados com seus estudos. Em Participar Prática, os estudantes e preceptores inserem, editam e visualizam conteúdos sobre pacientes. Em Participar Reunião, os estudantes e docentes participam de reuniões de Pequenos Grupos e, eventualmente, inserirem Síntese Provisória, Nova Síntese ou Questões de Aprendizagem. Os casos de uso Participar Reunião e Participar Prática possuem o estereotipo <<**context**>>, porque são influenciados pelo papel no processo de ensino (Docente, Estudante ou Preceptor) e pelo dispositivo do usuário. Em Gerenciar Prática, os preceptores podem acompanhar os atendimentos realizados por Estudantes. Em Gerenciar Reunião, facilitadores gerenciar problemas e visualizar os conteúdos criados durante as reuniões. Em

Gerenciar Pequeno Grupo, os facilitadores inserem estudantes, facilitadores, marcam Reuniões e inserem avaliações aos conteúdos criados por estudantes de um Pequeno Grupo. Em Acessar Conteúdo, os usuários acessam Síntese Provisória, Nova Síntese ou Questões de Aprendizagem. O contexto para este caso de uso é definido pelos atributos: localização, atividade e dispositivo do usuário; tipo (síntese provisória, nova síntese, questão de aprendizagem ou problema) e o idioma do conteúdo.

6.2. Análise

Os requisitos identificados para a Situação Problema foram analisados e suas especificações foram refinadas para tornar mais claro o que a aplicação deve atender.

Inicialmente, os comportamentos dos casos de uso são analisados considerando os tipos e seus atributos, representados em modelos de classes de análise dos objetos da aplicação. A Figura 6.2 apresenta um modelo de classes [OMG04] identificadas nessa fase do desenvolvimento. No caso, um PequenoGrupo é formado por um ou dois Docentes (Facilitador e Co-Facilitador) e um vetor de Estudantes. O PequenoGrupo realiza Reuniões, que possuem uma data, um local e um Problema. As Reuniões são especializadas em Práticas quando existe a participação de um Preceptor e de um ou mais Pacientes. Os conteúdos são relacionados com um PequenoGrupo, e são de autorias de Docentes, Estudantes ou Preceptores. O conteúdo foi especializado em Problema, Questões de Aprendizagem, Síntese Provisória, e Nova Síntese. As Questões de Aprendizagem possuem um vetor de Questões, que armazena separadamente cada questão inserida. A Nova Síntese pode ser avaliada por um Docente, caso seja necessário.

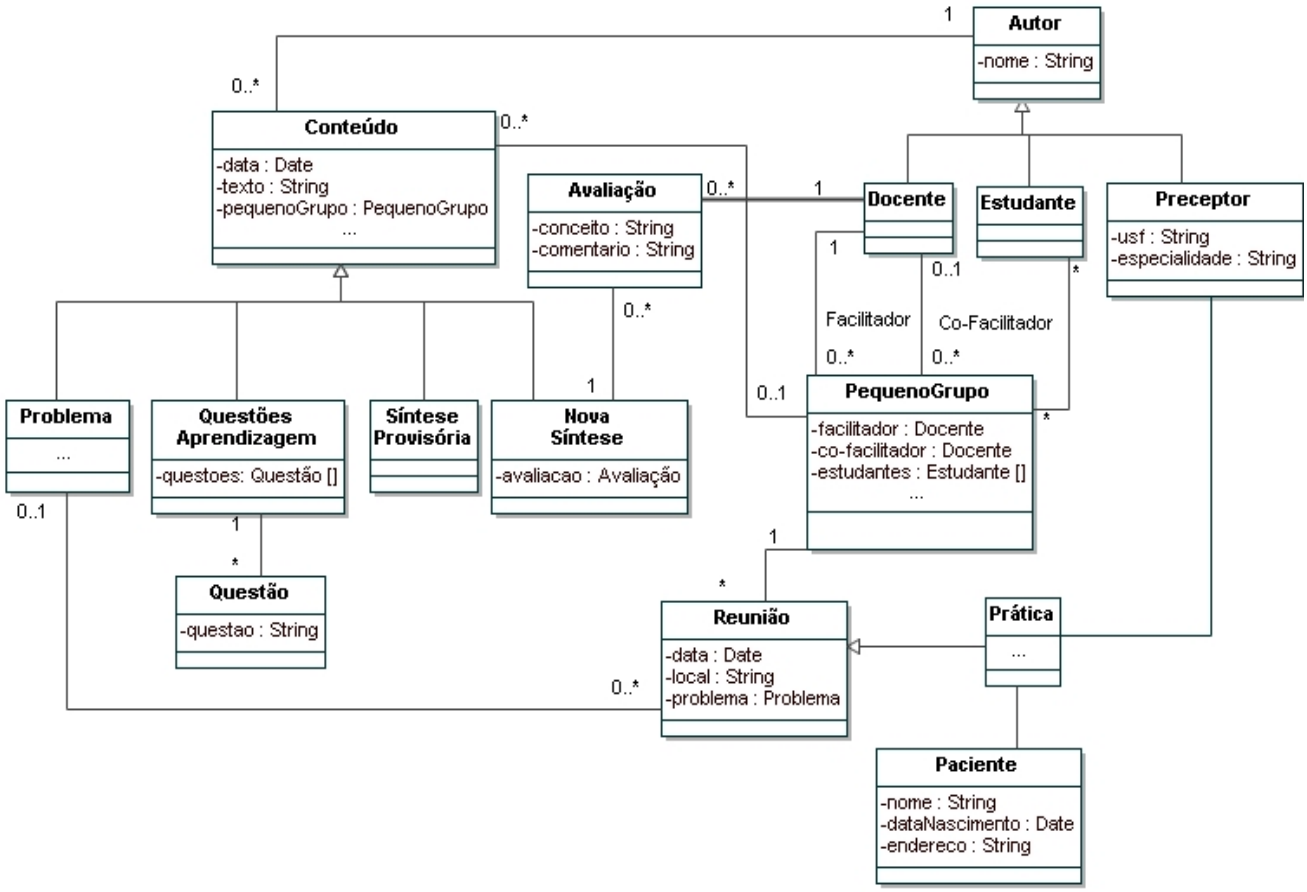

Figura 6.2. Modelo de Classes do PRE

Também são refinados os Requisitos Adaptáveis usando Ontologias. Assim, os casos de uso que são influenciados pelo contexto de uso da aplicação, são modelados em Ontologias. No PRE o contexto influencia o acesso aos conteúdos e a participação dos usuários em reuniões e práticas. A Figura 6.3 apresenta a Ontologia que descreve os requisitos adaptáveis para o PRE. No acesso aos conteúdos do PRE foram considerados: as localizações dos usuários, o dispositivo de acesso, os tipos dos conteúdos acessados (Síntese Provisória, Nova Síntese e Questões de Aprendizagem ou Problema), o idioma em que esses conteúdos foram escritos e a presença e o tamanho de figuras. Na participação dos usuários em reuniões e práticas são consideradas suas atividades (Situação Problema, Estações de Simulação, Prática

Profissional), o dispositivo de acesso e o papel desses usuários no processo de ensino e aprendizagem (Docente, Estudante ou Preceptor).

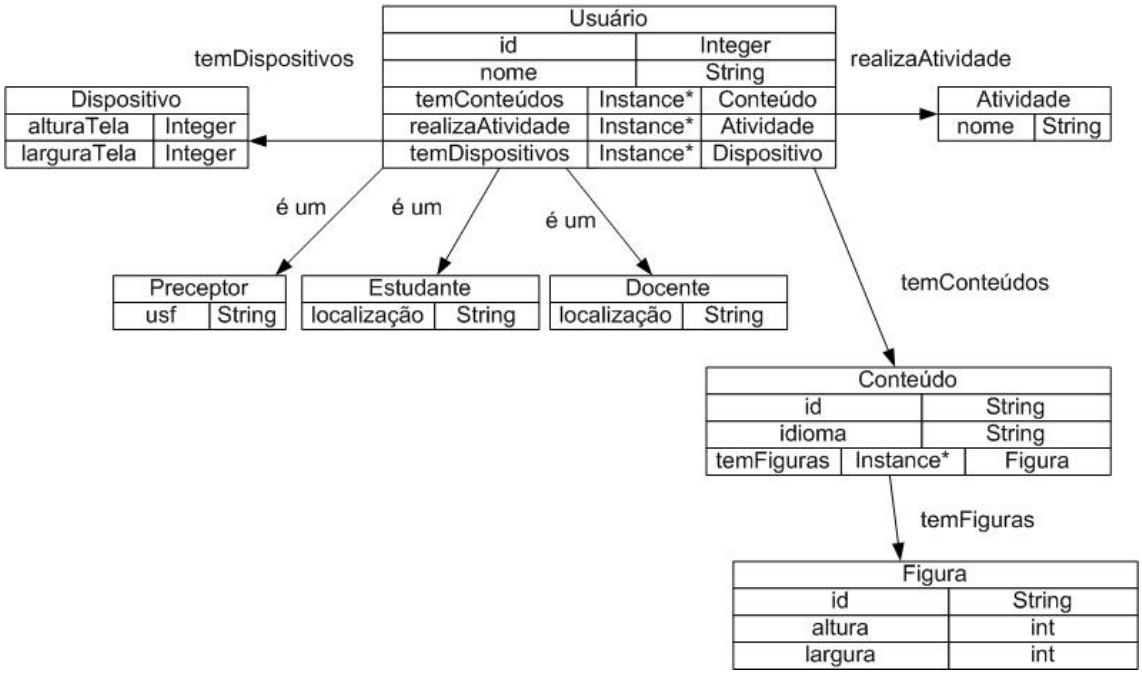

Figura 6.3. Modelo de Ontologias para Requisitos Adaptáveis do PRE

6.3. Projeto

A primeira atividade dessa disciplina foi definir a plataforma para desenvolvimento. No caso do PRE, foram tomadas as seguintes decisões de projeto, relativas à plataforma:*Java Micro Edition (JavaME)* [SUN] para construção dos Clientes móveis e *Java Enterprise Edition (JavaEE)* [SUNa] para os Clientes Desktop. Assim, foram desenvolvidos dois tipos de clientes: os móveis, em JavaME e os desktop, em JavaEE. Esses clientes poderão acessar servidores presentes: na UFSCar e em cada uma das Unidades de Saúde da Família.

A partir dessas decisões, a arquitetura da aplicação foi definida, conforme ilustra a Figura 6.4. No caso, um Docente usa um Desktop localizado na UFSCar conectado através da rede local, e um Estudante usa um Celular com a configuração CLDC através da Rede Telefônica enquanto estiver em

visita na família. Em seguida, o Estudante utiliza *Bluetooth* para descarregar as informações na UFS responsável pela família. O Preceptor utiliza um PDA conectado por wireless para acessar as informações coletadas pelo estudante. Ressalte-se que, os dispositivos de acesso, utilizados pelo Docente, Estudante e Preceptor, podem variar entre um Desktop, celular, PDA ou outro dispositivo.

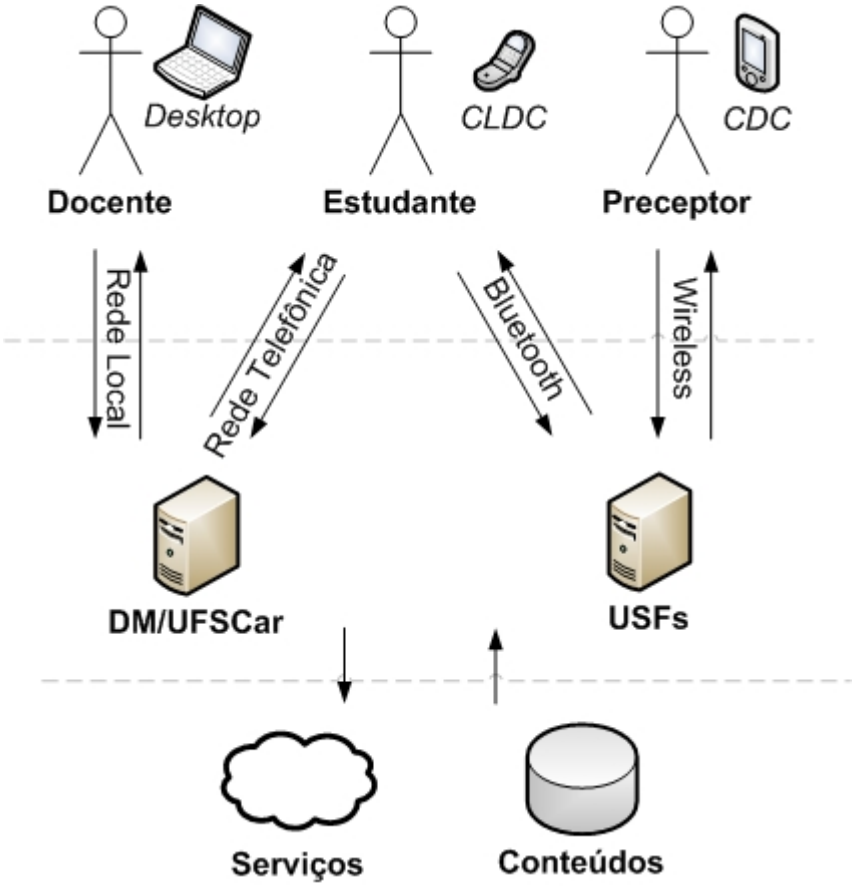

Figura 6.4. Arquitetura do PRE

Prosseguindo com o projeto, o contexto da aplicação é modelado através da extensão do modelo de Ontologias do UBICK, conforme mostra a Figura 6.5. As especificações da Ontologia, obtidas na disciplina de Análise, são refinadas e detalhadas fazendo reuso da Ontologia do UBICK conforme se segue: a classe Usuário estende a classe UserProfile, a Dispositivo estende a DeviceProfile e a Conteúdo estende a ContentProfile.

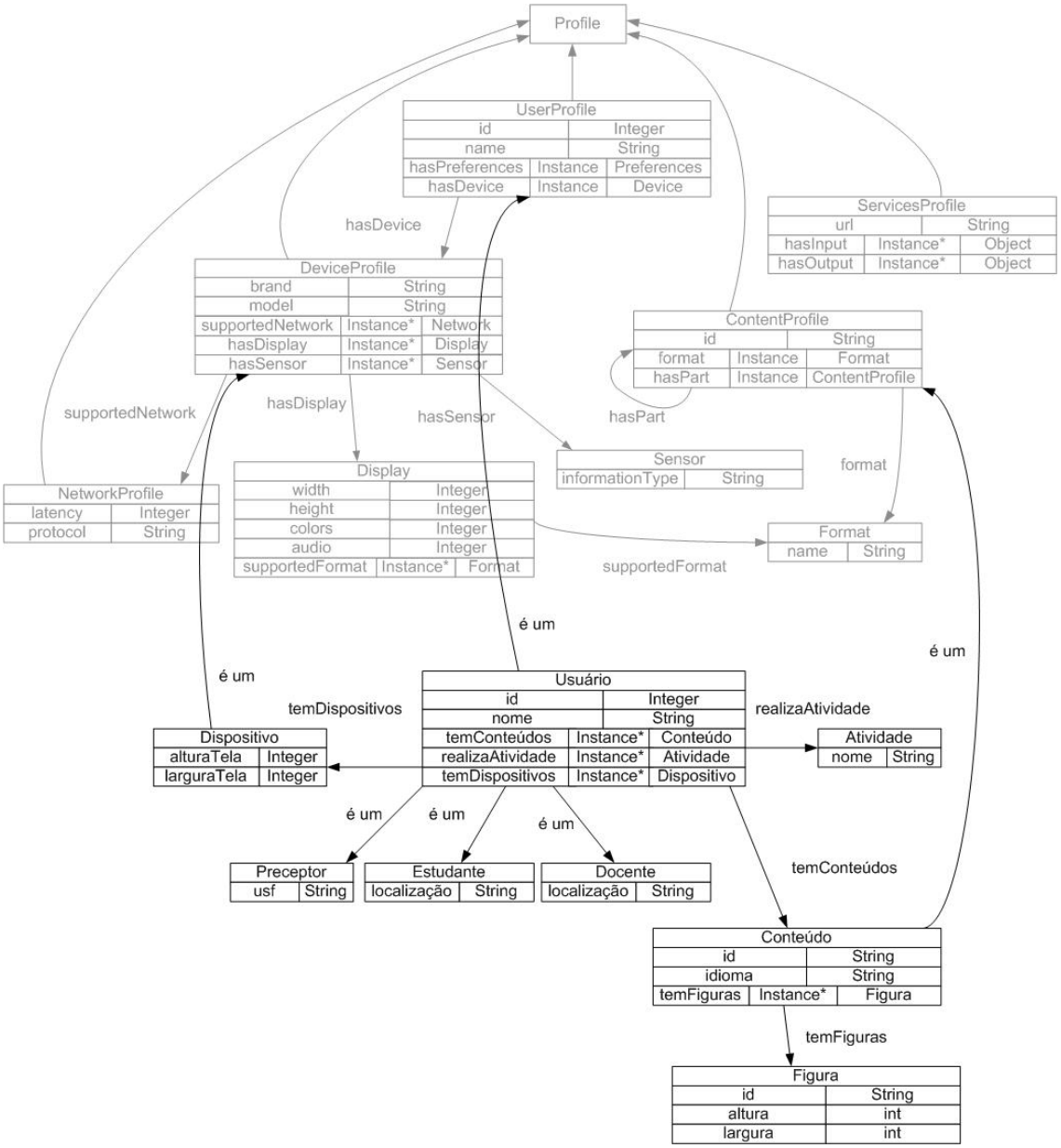

Figura 6.5. Extensão do Modelo de Ontologias do UBICK

Uma vez que o PRE reusa e refina a Ontologia do UBICK, se faz necessário que o mecanismo de inferência presente no componente BrokerAgent seja re-projetados. O novo mecanismo de inferências foi projetado para suportar regras de adaptação que considerem os requisitos adaptáveis específicos do PRE. Por exemplo, as regras para adaptações de conteúdo que desconsiderem figuras são: o usuário não estar localizado na Universidade ou

em uma USF; ou o usuário estar utilizando o PRE a partir de um celular conectado pela rede de telefonia.

A Figura 6.6 apresenta um modelo de componentes da parte cliente da aplicação, organizados nos pacotes PREMobile e PREDesktop, fazendo reuso dos componentes do UBICK. Os componentes do pacote PREMobile foram projetados segundo a tecnologia JavaME [Sun] e do pacote PREDesktop segundo a tecnologia *Java Server Faces (JSF)* [Sun]. O componente *Presentation* é responsável pela interface da aplicação. O componente *Authentication* faz a identificação do usuário mediante senha e e-mail. O *LocalizationListener* realiza leituras no GPS do dispositivo no PRE Mobile e informa o IP do Desktop no PRE Desktop. O *ActionListener* verifica qual é a ação do usuário no momento através do monitoramento da sua iteração com a interface do ambiente.

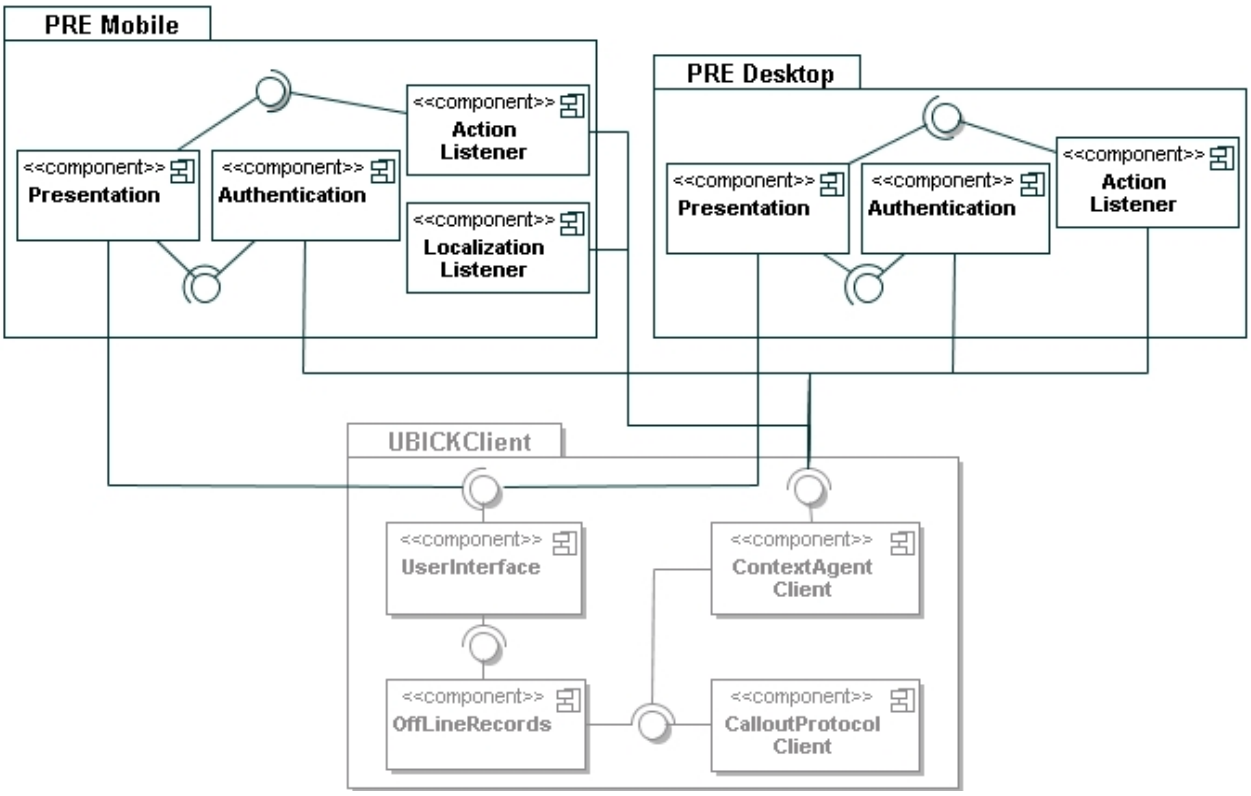

Figura 6.6. Modelo de componentes do PRE Cliente

A Figura 6.7 apresenta um modelo de componentes da parte Servidor da aplicação, organizados nos pacotes *SeverUFSCar* e *ServerUSF*, e fazendo reuso dos componentes do UBICK. O componente *Content* é responsável pela construção dos conteúdos da aplicação, cujas adaptações nos dispositivos ubíquos são realizadas pelo componente *ContentAdapter*. O componente *Groups* gerencia os Pequenos Grupos e suas Reuniões. Os componentes *Teachers* e *Students* são responsáveis pela criação e manutenção de Docentes e Estudantes, respectivamente. O componente *Pratics* é responsável pelas práticas, e pelo gerenciamento e manutenção de pacientes. O componente *HealthProfessionals* é responsável pela criação e manutenção dos Preceptores. Os componentes *Content*, *ContentAdapter* e *Students*, destacados com o estereótipo <<**webservice**>>, foram projetados como Serviços Web, para serem usados pelos dois servidores (*ServerUFSCar* e *ServerUSF*), evitando repetições.

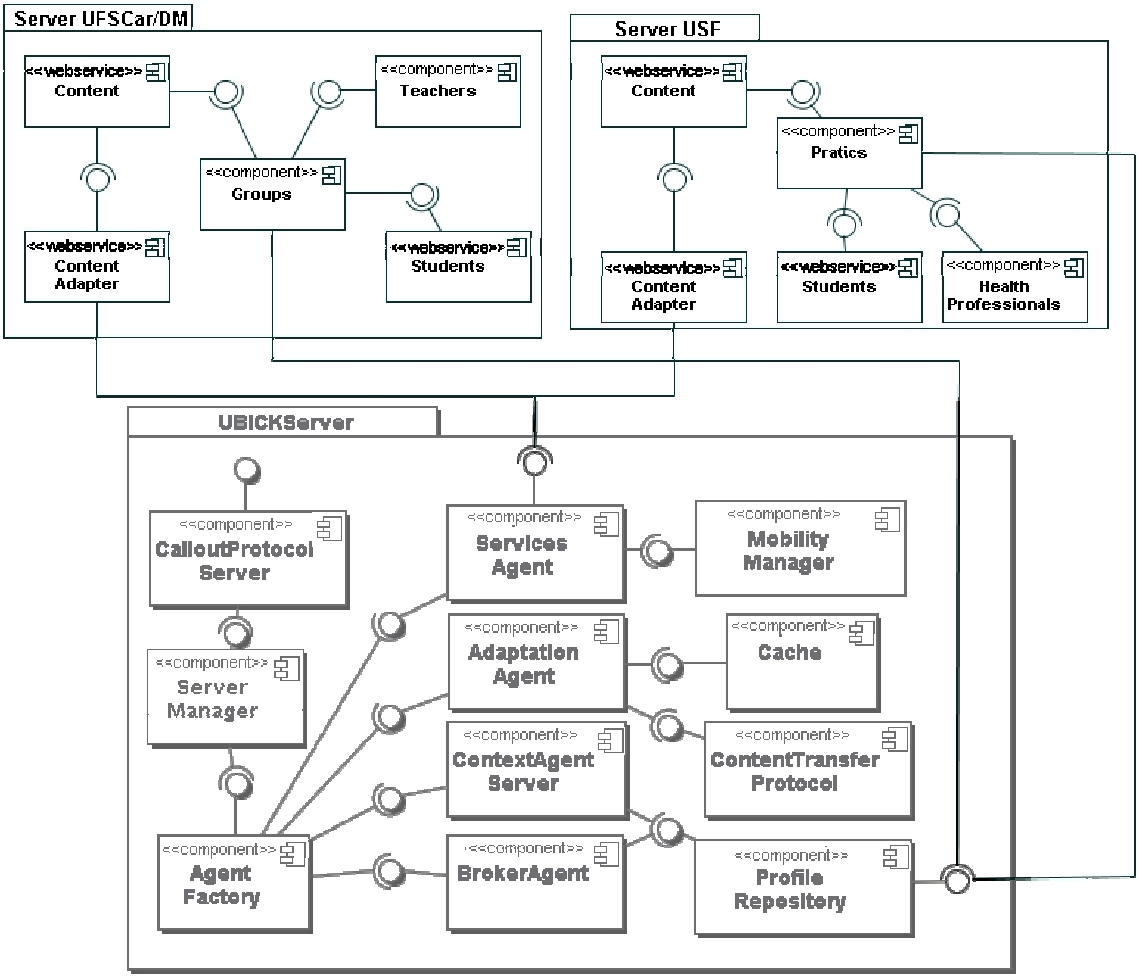

Figura 6.7. Modelo de componentes do PRE Servidor

6.4. Implementação

Prosseguindo com a abordagem foram implementados os componentes específicos da aplicação, fazendo reuso dos componentes do UBICK e dos componentes dos Serviços Web. No caso do PRE, os componentes Presentation, Authentication, LocalizationListener, ActionListener, Groups, Teachers, Pratics e HealtProfessional, foram implementados em versões para JavaME e JSF. Os componentes Content, ContentAdapter e Students foram implementados como Serviços Web.

Os componentes Students e Content são implementados como Serviços Web Semânticos com os métodos para realização do CRUD (Create, Retrieve, Update, e Delete). Durante a visualização pode ser verificado, a partir do

contexto, que um conteúdo deve ser adaptado. Para tal, o componente ContentAdapter foi transformado em três serviços Web: Conversor de Textos, que transforma textos da Internet em conteúdos acessível através de dispositivos móveis; Conversor de Imagens, que reduz e retira a qualidade de imagens presentes nesses textos; e Tradutor de Idioma, que traduz textos em inglês para português. Os dois primeiros serviços não estavam disponíveis, e, portanto, foram implementados; o último foi reutilizado a partir da ferramenta de tradução do Google[8].

Seguindo a abordagem AdeSCoU, foram especificadas as descrições OWL-S do serviços web. A Figura 6.8 apresenta uma dessas descrições, no caso para o serviço de tradução idiomas.

```
<!-- Service description -->
<service:Service rdf:ID="HTML2WMLService">
<service:presents rdf:resource="#HTML2WMLProfile"/>
<service:describedBy rdf:resource="#HTML2WMLProcess"/>
<service:supports rdf:resource="#HTML2WMLGrounding"/>
</service:Service>

<!-- ServiceProfile description -->
<profile:HTML2WMLService rdf:ID="HTML2WMLrProfile">
<service:presentedBy rdf:resource="#HTML2WMLService"/>
<profile:serviceName>HTML2WML</profile:serviceName>
<profile:hasInput rdf:resource="#htmlpage"/>
<profile:hasOutput rdf:resource="#wmlpage"/>
</profile:HTML2WMLService >

<!-- ServiceProcess description -->
<process:AtomicProcess rdf:ID="HTML2WMLProcess">...

<!-- ServiceGrounding description -->
<grounding:WsdlGrounding rdf:ID="HTML2WMLGrounding">
<service:supportedBy rdf:resource="#HTML2WMLService"/>
</grounding:WsdlGrounding>
<grounding:WsdlAtomicProcessGroundingr df:ID="HTML2WMLProcessGrounding">
<grounding:owlsProcess rdf:resource="HTML2WMLProcess"/>
<grounding:wsdlDocument>
http://localhost/HTML2WML/HTML2WMLService?wsdl
</grounding:wsdlDocument>
```

Figura 6.8. Descrição de um Serviço Web Semântico

[8] *http://www.google.com/translate_t*

6.5. Testes

Para a realização de Testes, o protótipo do PRE Web foi tornado disponível no ambiente de uso, conforme ilustra a Figura 6.9. Foi constituído um Grupo Piloto formado por um docente e seis estudantes do segundo ano do Curso de Medicina da UFSCar. As reuniões desse grupo ocorreram duas vezes por semana, durante todo o 2º semestre de 2007, num ambiente que contou com um laptop para cada participante do grupo e câmeras e microfones para o registro dessa atividade.

Figura 6.9. Uso do PRE

O gráfico da Figura 6.10 apresenta a quantidade de documentos criados no decorrer das semanas, onde um documento representa qualquer informação inserida pelo usuário que seja relevante ao problema tratado (e.g., imagens, pesquisas, o próprio problema). De acordo com esse gráfico, após o pico na 2ª semana, possivelmente ocasionado por testes e pela inexperiência

dos usuários, ocorreu um crescimento gradual até a 7ª semana, sendo interrompido na 8ª semana devido a um recesso na UFSCar. A partir da 9ª semana, o crescimento gradual voltou a ser observado.

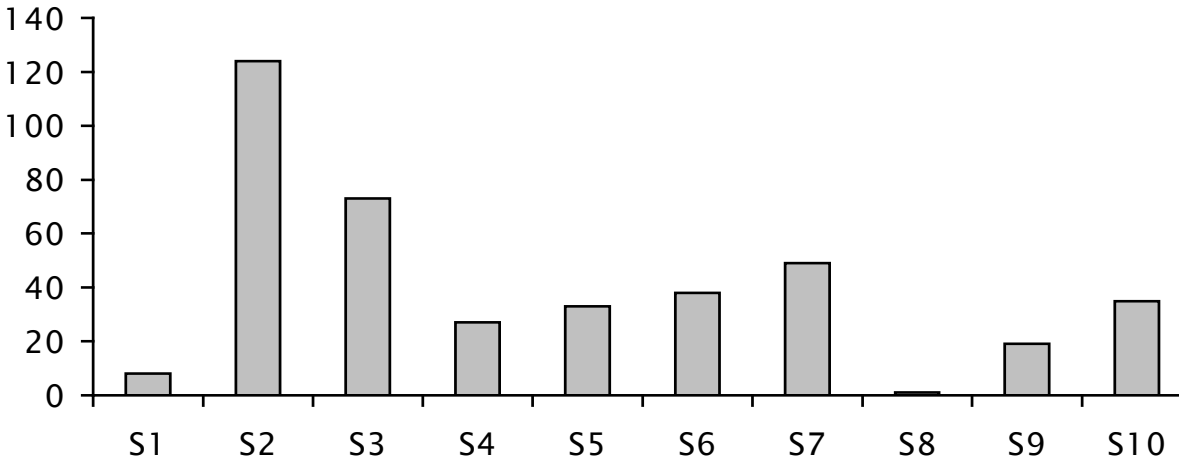

Figura 6.10. Documentos Criados no PRE

O gráfico da Figura 6.11 mostra que muitos documentos criados não foram compartilhados. Da interação dos desenvolvedores do PRE com seus usuários estudantes, ficou esclarecido que essa tarefa não ocorreu por opção do estudante ou por seu desconhecimento sobre o compartilhamento. Os resultados obtidos ao longo dos testes serviram para corrigir e melhorar o PRE e para confirmar a necessidade de um melhor treinamento dos seus usuários.

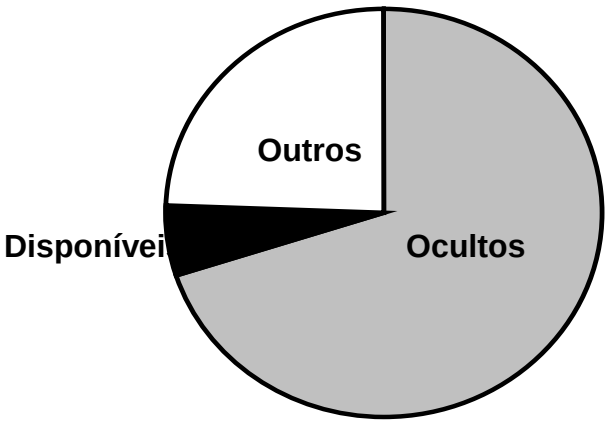

Figura 6.11. Estados dos Documentos em Relação ao Compartilhamento

O gráfico da Figura 6.12 mostra os documentos em relação a suas versões. A área denominada “Finais com Versões” representa documentos finais que evoluíram a partir de versões anteriores. A área denominada “Finais sem Versões” representa documentos finais que foram criados e não mais editados. Os documentos representados pelas áreas anteriores, seriam armazenados em portfólios tradicionais de papel, mas a grande quantidade de documentos, representada na área denominada “Versões”, que refletem a evolução do estudante durante suas pesquisas, seria descartada sem uso do PRE.

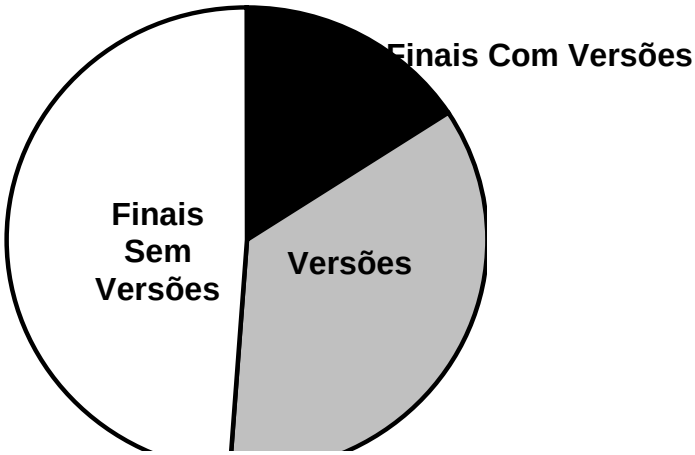

Figura 6.12. Documentos e Versões

Baseado nos testes o PRE foi refinado até a versão atual que implementa o processo ABP de ensino e aprendizagem nas atividades curriculares Situação Problema, Estações de Simulação e Prática Profissional. Em particular, na atividade curricular Prática Profissional, os estudantes e professores em ambiente não protegido (fora da universidade), podem acessar o PRE via dispositivos móveis ou através de laptops. Por exemplo, a Figura 6.13 ilustra a adaptação de um documento do PRE para telefones celulares que suportam *Mobile Information Device Profile (MIDP)* do JavaME [SUN].

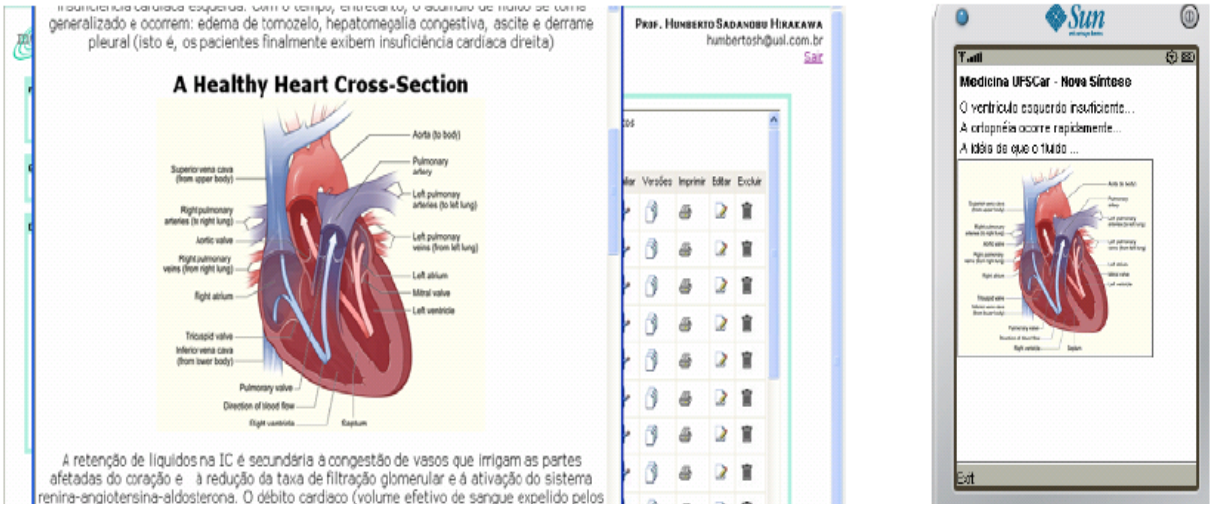

Figura 6.13. PRE Desktop e PRE Mobile

6.6. Implantação

O PRE ainda não foi implantado para produção. A implantação ocorreu apenas para a realização dos testes com o grupo piloto do curso de Medicina. Nessa versão de testes o *deployment* do PRE foi realizado no servidor TomCat versão 6 e com o Servidor de Banco de Dados Postgres versão 8.3.

Atualmente, um novo ciclo do processo de desenvolvimento está sendo realizado para produzir um ambiente mais completo, que suporte outras funcionalidades do processo de ensino e aprendizagem. Essa nova versão do PRE tem apoio do Departamento de Medicina e do Grupo de Computação Ubíqua da UFSCAR. Para que essa nova versão entre em produção são necessários criar materiais de apoio, e, possivelmente, realizar sessões de treinamentos para os envolvidos.

6.7. Conclusões

Este capítulo apresentou o desenvolvimento de uma aplicação ubíqua que auxilia o processo de ensino e aprendizagem no curso de Medicina. A

aplicação foi desenvolvida com base nos requisitos de um Portfólio Reflexivo Eletrônico, que auxilia o ensino dos estudantes do curso de Medicina.

Os principais benefícios oferecidos por aplicações para aprendizado ubíquo [CHE08], semelhantes à apresentado nesse estudo de caso, incluem: disponibilidade das informações disponíveis “a qualquer hora, em quaisquer lugares”, e o uso de dispositivos para anotação de idéias, acompanhamento da construção do conhecimento, compartilhamento de conhecimento, e eliminação do papel. O protótipo construído do PRE pode ser melhorado para incluir novos serviços para adaptação de conteúdo (e.g., vídeo e áudio), e a integração do PRE com prontuários eletrônicos da rede de saúde.

Capítulo 7

Conclusões e Trabalhos Futuros

Esta dissertação apresentau os resultados de uma pesquisa sobre uma Abordagem para o Desenvolvimento de Software na Computação Ubíqua, denominada AdeSCoU.

No Capítulo 4, foi apresentado o desenvolvimento do UBIquitous Computing frameworK (UBICK). Este framework tem como principais requisitos: Ciência de Contexto, Adaptabilidade, Adaptação de Conteúdo, Onipresença, Descoberta e Composição de Serviços, Interoperabilidade, Heterogeneidade e Tolerância à Falhas. Para implementar os componentes do UBICK, foram combinadas Ontologias, Serviços Web Semânticos e Agentes de Software. Ontologias facilitaram a especificação e uso do contexto das aplicações ubíquas. Serviços Web Semânticos facilitam a onipresença, descoberta, composição e uso de serviços nas aplicações desenvolvidas com a AdeSCoU. Agentes de Software gerenciam as aplicações desenvolvidas através de reuso do UBICK, proporcionando adaptabilidade e facilitando a implementação de Serviços.

No Capítulo 5 foi apresentada a abordagem AdeSCoU, uma extensão do Rational Unified Process (RUP), que emprega o UBICK para simplificar grande parte da implementação de aplicações ubíquas. O RUP facilitou o desenvolvimento da AdeSCoU, como sua instância, especializada nas suas

fases e disciplinas, considerando as características específicas da Computação Ubíqua.

No Capítulo 6 foi apresentado um estudo de caso que ilustra o uso da AdeSCoU, para o desenvolvimento do Portfólio Reflexivo Eletrônico, a partir de requisitos identificados no domínio de educação do curso de Medicina da Universidade Federal de São Carlos. O exemplo destaca os principais passos da abordagem e o reuso do framework UBICK. Dentre os requisitos ubíquos do PRE, destacam-se: disponibilidade das informações disponíveis “a qualquer hora, em quaisquer lugares”, compartilhamento de conhecimento e eliminação do papel, e a adaptações dos diferentes conteúdos.

7.1. Dificuldades e Limitações

As principais dificuldades encontradas foram devidas ao conhecimento ainda embrionário da Engenharia de Software na Computação Ubíqua, suas constantes mudanças, e as limitações de tempo.

Uma das dificuldades relaciona-se com a utilização de tecnologias não consolidadas, ou em fase de consolidação, principalmente em relação à Web Semântica. Poucos frameworks disponibilizam componentes eficientes para trabalhar com Ontologias e Serviços Web Semânticos. Assim, foi necessário estender o UBICK para suportar a criação de algoritmos específicos para descoberta e composição de Serviços para Adaptação de Conteúdo.

Outras limitações importantes da AdeSCoU incluem: inexistência de componentes e atividades específicas para a criação de interfaces e padrões de Interface Humano Computador (IHC); e ausência de um estudo de caso que considere o uso dessa abordagem em ambientes inteligentes.

7.2. Recomendações para Trabalhos Futuros

A Computação Ubíqua é complexa e envolve vários conceitos, idéias, e tecnologias, que podem ser exploradas para refinar a abordagem AdeSCoU proposta.

No UBICK, por exemplo, podem ser realizadas as seguintes melhorias: utilizar-se Ontologias Fuzzy para especificar e representar contextos que indiquem situações de incerteza em aplicações ubíquas; considerar-se outros aspectos dos Serviços Web como qualidade de serviço, utilização de BPEL4WS e controle de exceção; e criar-se novos Agentes de Software, para melhoria da comunicação entre os agentes já existentes no UBICK e maior mobilidade dos Agentes de Software entre clientes e servidores.

Na AdeSCoU, trabalhos futuros incluem a criação de ferramentas que auxiliem os Engenheiros de Software na execução de suas diferentes atividades. Assim, tem-se a extensão da abordagem com as disciplinas de Modelagem de Negócios, Gerenciamento de Configuração, Gerenciamento de Projeto e Ambiente. Essas disciplinas beneficiam com o aumento de produtividade, melhor entendimento do impacto da ubiqüidade em relação aos negócios de software e melhor gerenciamento das modificações nos requisitos.

Finalmente, em relação ao PRE, podem ser incluídos novos serviços para adaptação de conteúdo (e.g., vídeo e áudio) e para sua integração com prontuários eletrônicos da rede de saúde.

Referências

[ABO99] Aboud, G. “Software Engineering Issues for Ubiquitous Computing” ACM Computing Surveys, vol. 28, no. 4, 1999.

[AIT07] Aitenbichler, E., Kangasharju, J. e Mühlhäuser, M. “MundoCore: A light-weight infrastructure for pervasive computing” Pervasive Mobile Computing, vol. 3, no. 4, pp. 332-361, 2007.

[BAL06] Balasubramanian, M., Chaturvedi, N., Chowdhury, A. e Ganesh, A. “A framework for rapid-prototyping of context based ubiquitous computing applications” Anais da Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, pp. 306-311, 2007.

[BAR07] Barbosa, J., Hahn, R., Barbosa, D. e Geyer, C. “Mobile and Ubiquitous Computing in an Innovative Undergraduate Course” ACM SIGCSE Bulletin, vol. 39, no. 1, pp. 379-383, 2007.

[BEL07] Bell G. e Dourish P. “Yesterday’s tomorrows: notes on ubiquitous computing’s dominant vision”c Personal and Ubiquitous Computing, no. 11, pp. 133–143, 2007.

[BEL08] Bellifemine, F., Caire, G., Poggi, A. e Rimassa, G. “JADE: A software framework for developing multi-agent applications. Lessons learned” Information and Software Technology, vol. 50, no. 1-2, pp. 10-21, 2008.

[BER01] Berners-Lee, T., Hendler, J. e Lassila, O. ”The Semantic Web” Scientific American, vol. 5, no. 17, pp. 35-43, 2001.

[BER05] Berry, D., Cheng, B. e Zhang, J. "The four levels of requirements engineering for and in dynamic adaptive systems" Anais do Workshop on Requirements Engineering: Foundation for Software Quality, 2005.

[BLI95] Bligh, J. “Problem-based learning in medicine: an introduction”, Postgrade Medicine Journal, no. 71, pp. 323-326, 1995.

[BUL06] Bulcão, R. “Um processo de software e um modelo ontológico para apoio ao desenvolvimento de aplicações sensíveis à contexto”. Tese de Doutorado, Universidade de São Paulo – São Carlos, 2006.

[BUL07] Bulcão, R., Kudo, N. e Pimentel, M. “POCAp: A Software Process for Context-Aware Computing” Anais do Conference on intelligent Agent Technology, pp. 705-708, 2007.

[CAR02] Cardoso, S., Kon, F. “Mobile Agents: A Key For Effective Pervasive Computing” Anais do Workshop on Pervasive Computing, 2002.

[CAR07] Carton, A., Clarke, S., Senart, A. e Cahill, V. “Aspect-Oriented Model-Driven Development for Mobile Context-Aware Computing” Anais da International Conference on Software Engineering Workshops, pp. 5-5.

[CHA06] Charif-Djebbar Y. e Sabouret N. (2006) “Dynamic Service Composition and Selection through an Agent Interaction Protocol”. Anais da International Conference on Web Intelligence and Intelligent Agent Technology, pp. 105-108.

[CHE04] Chen, H. e Finin, T. “An Ontology for Context Aware Pervasive Computing Environments” The Knowledge Engineering Review, vol. 18, no. 3, pp. 197-207, 2004.

[CHE08] Chen, G.D., Chang, C.K. e Wanga, C.Y. “Ubiquitous learning website: Scaffold learners by mobile devices with information-aware techniques”, Computers & Education, vol. 50, no. 1, pp. 77-90, 2008.

[CHR05] Christopoulou E. e Kameas A. “GAS Ontology: An ontology for collaboration among ubiquitous computing devices” International Journal of Human-Computer Studies, vol. 62, no. 5, pp. 664-685, 2005.

[CLA05] Claudino, R. “Uma arquitetura baseada em componentes para adaptação de conteúdo na Internet”. Dissertação de Mestrado - Departamento de Computação, Universidade Federal de São Carlos, 77p,, 2005.

[CON05] Conselho de Ensino e Pesquisa da Universidade Federal de São Carlos. (2005) “Curso de Medicina da UFSCar”. <www2.ufscar.br/graduacao/medicina.php\>.

[COO06] Coordenação da Graduação em Medicina do Centro de Ciências Biológicas e da Saúde da Universidade Federal de São Carlos, “Caderno do Curso – Ciclo I – 1º ano”, Universidade Federal de São Carlos, 78 p, 2006.

[DEY01] Dey, A. “Personal and Ubiquitous Computing Journal”, Volume 5 (1), 2001, pp. 4-7.

[END05] Endres C. “A survey of software infrastructures and frameworks for ubiquitous computing, mobile information systems”, vo.1, no. 1, pp. 41-80, 2005.

[FAL07] S Fallis, R Payne, R Limb e D Allison, “Pervasive information — the key to ‘true’ mobility”, BT Technology Journal, vol. 25, no. 2, 2007

[FER07] Fernandes, J., Machado, R., Carvalho, J. “Model-Driven Software Development for Pervasive Information Systems Implementation” Anais do International Conference on Quality of Information and Communications, pp. 218-222, 2007

[FOR07] Forte, M., Souza, W. e Prado, A. “Using ontologies and Web services for content adaptation in Ubiquitous Computing”, Journal of Systems and Software, doi:10.1016/j.jss.200.04.044, 2007.

[FOR06] Forte, M.. “Especificação de perfis e regras, baseada em Ontologias, para adaptação de conteúdo na Internet” Dissertação de Mestrado, Universidade Federal de São Carlos, 150p., 2006.

[FRA96] Franklin, S. e Graesse, A., “Is it an agent, or just a program? A taxonomy for autonomous agents” Anais of the Third International Workshop on Agent Theories, Architecture and Languages, 1996.

[GAM95] Gamma E., Helm R., Johnson R. e Vlissides J “Design Patterns: Elements of Reusable Object-Oriented Software”. Addison-Wesley, 1995.

[GUI05] Guizzardi, G. “Ontological Foundations for Structural Conceptual Models” Tese de doutorado, Universidade de Twente, 2005.

[HAN03] Hansmann, U., Stober, T., Merk, L. e Nicklous, M. “Pervasive Computing”, Springer-Verlag, Second Edition, 2003.

[HEL05] Helal, S. “Programming Pervasive Spaces”, Pervasive Computing, vol. 1, no. 1, pp. 84-87, 2005.

[IBM] RUP SOA Plugin. Diponível em: <http://www.ibm.com/developerworks/rational/downloads/06/rmc_soma/>. Último acesso em Fevereiro de 2008.

[IBMa] RUP Governance Plugin. Disponível em: <http://www.ibm.com/developerworks/rational/downloads/06/plugins/rmc_soa_gov/soa_plugin.html>. Último acesso em Fevereiro de 2008.

[JAD] JADE - Java Agent DEvelopment Framework. Disponível em <http://jade.tilab.com/>. Último acesso em Fevereiro de 2008.

[JEN] Jena – A Semantic Web Framework for Java. Disponível em <http://jena.sourceforge.net/>. Último acesso em Fevereiro de 2008.

[JEN98] Jennings, N., Sycara, K. e Wooldridge, M. “A Roadmap of Agent Research and Development, Autonomous” Agents and Multi-Agent Systems, vol. 1, no. 1, pp. 7 - 38, 1998.

[JEN99] Jennings, N. e Wooldridge, M. “Agent-Oriented Software Engineering”, Anais do 9th European Workshop on Modeling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering, pp. 1-7, 1999.

[JUN] JUnit. Disponível em <http://www.junit.org/>. Último acesso em Fevereiro de 2008.

[KIM06] Kim, Y., Kim, E., Kim, J., Song e E., Ko, I. “Ontology Based Software Reconfiguration in a Ubiquitous Computing Environment” Anais da IEEE International Conference on Computer and Information Technology, pp. 260, 2006.

[KIN02] T. Kindberg e A. Fox, "System Software for Ubiquitous Computing," IEEE Pervasive Computing, pp. 71-80, 2002.

[KIT04] Kitchenham, B. “Procedures for Performing Systematic Reviews”, Relatório técnico, Grupo de Engenharia de Software, Departamento de Ciência da Computação, Universidade de Keele, Austrália, 2004.

[KRI05] Krikke, J. “Pervasive Computing T-Engine: Japan’s Ubiquitous Computing Architecture Is Ready for Prime Time”, IEEE Pervasive Computing, vol. 4, no. 2, pp. 1536-1268, 2005.

[KRU03] Kruchten, P. “Rational Unified Process, The: An Introduction, Third Edition”, Addison Wesley., 2003.

[KUL07] Kulkarni, D. e Tripathi “Generative Programming Approach for Building Pervasive Computing Applications” Anais da International Conference on Software Engineering Workshop, pp. 189.

[LAN99] Lange, D. e Oshima, M. “Seven Good Reasons for Mobile Agents”, Communications of ACM, vol. 42, no. 3, pp. 88-89, 1999.

[LAN01] Lamsweerde, A. “Goal-Oriented Requirements Engineering: A Guided Tour” Anais do Fifth IEEE international Symposium on Requirements Engineering pp. 249.

[LEE07] Lee, W. “Deploying personalized mobile services in an agent-based environment”, Expert Systems with Applications, vol. 32, no. 4, pp. 1194-1207, 2007.

[MAR03] Martin D. et al. OWL-S: Semantic Markup for Web Services 1.0. OWL-S Coalition, 2003. Disponível em: <http://www.daml.org/services/owl-s/1.0/> Último acesso em Fevereiro de 2008.

[MAR08] Martins, D.S., Santana, L. H. Z., Biajiz, M., Prado, A. F., Souza, W.L.. “Context-Aware Information Retrieval On An Ubiquitous Medical Learning Environment” Anais do Symposium On Applied Computing, a ser publicado, 2008.

[MCG04] McGuinness, D. L. and Harmelen, F. V. “OWL Web Ontology Language Overview”, W3C, 2004. disponível em <http://www.w3.org/TR/owl-features> Último acesso em Fevereiro de 2008.

[MIN] Mindswap OWL-S API. Disponível em <http://www.mindswap.org/2004/owl-s/index.shtml> Último acesso em Fevereiro de 2008.

[MIN07] Min, X., Jizhong, Z. Yong, Q., Hui, H, Ming, L. e Wei, W. “Isotope Programming Model: a Kind of Program Model for Context-Aware Application” Anais da International Conference on Multimedia and Ubiquitous Engineering, pp. 597-602, 2007.

[NIE04] Niemelä, E. e Latvakoski, J. “Survey of Requirements and Solutions for Ubiquitous Software” Anais da International conference on Mobile and ubiquitous multimedia, pp. 71 - 78 , 2004

[NIG05] Niguidula, D., Ring, G. e Davis, H. (2005) “Digital Portfolios: A Dozen Lessons in a Dozen Years”. Disponível em <www.richerpicture.com/dozenLessons.pdf>. Último acesso em Fevereiro de 2008.

[OAS] OASIS, UDDI White Paper, 2004. Disponível em <http://uddi.org/whitepapers.html>. Último acesso em Fevereiro de 2008.

[OGL07] Ogliari, S., Cervi, R. e Dorneles, F. “Uma Arquitetura para Comunicação Móvel em Meios Intermitentes Utilizando Dispositivos com Recursos Computacionais Limitados” Anais do Simpósio Brasileiro em Multimídia e Web, pp. 267-273, 2007.

[OMG04] OMG. (2004) Unified Modeling Language (UML) Specification, Version 2.1.1, Object Management Group.

[IPP05] Pandis, I. Soldatos, J. Paar, A. Reuter, J. Carras, M. e Polymenakos, L. “An Ontology-based Framework for Dynamic Resource M anagement in Ubiquitous Computing Environments” Anais da International Conference on Embedded Software and Systems, pp. 195 – 203, 2005.

[PAP03] Papazoglou, M.P. “Service-oriented computing: concepts, characteristics and directions” Anais da Conference on Web Information Systems Engineering, pp. 3–12, 2003.

[PER07] Perlin, C., Santana, L., Martins, D., Prado, A., Souza, W., Biajiz, M. . “Um Ambiente De Computação Ubíqua Para O Ensino Baseado Em PBL” Anais Da Xxxiv Conferencia Latinoamericana De Informática, 2007.

[PHA07] Pham, N., Mahmoud, H., Ferworn, A. e Sadeghian, A. “Applying Model-Driven Development to Pervasive System Engineering” Anais do Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments, pp. 7, 2007.

[PRE02] Pressman, R. S. “Engenharia de Software” 5º Edição McGrawHill

[RIV06] Riva, O., Di Flora, C., Russo, S. e Raatikainen, K. “Unearthing Design Patterns to Support Context-Awareness” Anais da IEEE international Conference on Pervasive Computing and Communications, pp. 383, 2006.

[SAH03] Saha, D. e Mukherjee, A. 2003. Pervasive Computing: A Paradigm for the 21st Century. Computer 36, 3 (Mar. 2003), 25-31. DOI= http://dx.doi.org/10.1109/MC.2003.1185214

[SAN07] Santana, L., Martins, D., Forte, M., Souza, W., Prado, A., Biajiz, M. e Knoff, L. “Serviço De Tradução De Linguagens De Marcação Para A Internet”. Anais Do Simpósio Brasileiro De Redes De Computadores, 2007. pp. 541-554.

[SAN07a] Santana, L. H. Z. ; Martins, D.S. ; Perlin, C. B. ; Prado, A. F. ; Souza, W.L.; Biajiz, M. “Adaptação De Páginas Web Para Dispositivos Móveis”. Simpósio Brasileiro De Sistemas Multimídia E Web, pp. 1-8, 2007.

[SAN07b] Santana, L, Prado, A., Souza, W., Biajiz, M. “Usando Ontologias, Serviços Web Semânticos E Agentes Móveis No Desenvolvimento Baseado Em Componentes”. Simpósio Brasileiro De Componentes, Arquiteturas E Reutilização De Software, pp 163-176, 2007.

[SAN07c] Santana, L., Prado, A. , Souza, W., Biajiz, M. “Uma Abordagem Para A Adaptação De Conteúdo Da Internet”. Anais Do Ongoing Thesis And Dissertations. P. 206-210., 2007.

[SAN08] Santos, H.F., Santana, L.H.Z., Martins, D.S., Souza, W.L.; Prado, A.F., Biajiz, M. “An Ubiquitous Environment for Medical Education” Anais do Symposium on Applied Computing, a ser publicado, 2008.

[SAT01] Satyanarayanan, M (2001) “Pervasive Computing: Vision and Challenges” IEEE Personal Communications, vol. 8, no. 4, pp. 10-17, 2001.

[SER98] Serugendo, G., Muhugusa, M. e Tschudin, C. “A survey of theories for mobile agents” World Wide Web, vol. 1, pp. 139–153, 1998.

[SOL07] Soldatos, J., Pandis, I., Stamatis, K., Polymenakos, L., Crowley, J. “Agent based middleware infrastructure for autonomous context-aware ubiquitous computing services”, Computer Communications, vol. 30, no. 3, pp. 577-591, 2007.

[SOR07] Soroker, D., Cáceres, R., Dig, D., Schade, A., Spraragen, S., e Tiwari, A. “Pegboard: a framework for developing mobile applications” Anais da International Conference on Mobile Systems, Applications and Services, pp. 138 – 150, 2007.

[SPI07] Spínola, R., Massollar, J., Travassos, G. (2007) “Checklist to Characterize Ubiquitous Software Projects” Anais do Simpósio Brasileiro de Engenharia de Software, pp. 39-55.

[SUN] Java Micro Edition. Disponível em: <http://java.sun.com/javame/index.jsp> Último Acesso em Fevereiro de 2008.

[SUNa] Java Enterprese Edition. Disponível em: <http://java.sun.com/javaee/> Último Acesso em Fevereiro de 2008.

[SUNb] Java Server Faces. Disponível em: <http://java.sun.com/javaee/javaserverfaces/> Último Acesso em Fevereiro de 2008.

[SUY06] Syukur, E. e Wai Loke, S. “The MHS Methodology: Analysis and Design for Context-Aware Systems” Anais do IEEE Workshop on Software Technologies For Future Embedded and Ubiquitous Systems, and the Second international Workshop on Collaborative Computing, integration, and Assurance, pp. 11-16, 2006.

[TOR04] Torres, S. “Uma Linguagem de Modelagem para Aplicações Multi-agentes Baseada num Framework Conceitual para Agentes e Objetos” Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 252p., 2004.

[TOR07] Torres, S. e Lucena, C. “Modeling multi-agent systems”Comunications of the ACM, vol. 50, no. 5, pp. 103 – 108, 2007.

[W3C04] W3C, OWL Web Ontology Language Guide, 2004.

[W3C01] W3C, Web Services Description Language (WSDL) 1.1. Diponível em: <http://www.w3.org/TR/wsdl> Último Acesso em Fevereiro de 2008.

[WAN05] Want, R. e Pering, T. “System challenges for ubiquitous & pervasive computing” Anais da International conference on Software engineering, pp. 9 – 14.

[WEI91] M. Weiser “The Computer for the Twenty-First Century” Scientific American, vol. 265, no. 3, pp. 94-104, 1991.

[WEI94] M. Weiser, “The world is not a desktop” ACM Interactions vol. 1, no.1, pp. 7-8, 1994.

[WU08] Wu, S., Chang, C., Ho, S., e Chao, H. 2008. “Rule-based intelligent adaptation in mobile information systems” Experts Systems Application, vol. 34, no.2, pp. 1078-1092, 2008.

[ZAM04] Zambonelli, F. e Omicini, A. “Challenges and Research Directions in Agent-Oriented Software Engineering”, Journal of Autonomous Agents and Multiagent Systems, vol. 9, no. 3, pp. 253-284, 2004.

[YIJ07] Yu, Yijun; Lu, Jianguo; Fernandez-Ramil, Juan; Yuan, Phil “Comparing Web Services with other Software Components” Anais da IEEE International Conference on Web Services, pp. 388-397.

Publicações

A partir da pesquisa apresentada nessa dissertação, foram publicados, em colaboração com diferentes pesquisadores, os seguintes artigos em eventos nacionais e internacionais:

Forte, M.; Claudino, R.A.T.; Prado, A. F. ; Souza, W.L.; Santana, L. H. Z. . A Component-Based Framework For The Internet Content Adaptation Domain. Anais do Symposium on Applied Computing 2007 (Sac07), 2007. V. 2. P. 1450-1455.

Santana, L. H. Z. ; Martins, D.S. ; Forte, M. ; Souza, W.L. ; Prado, A. F. ; Biajiz, M. ; Knoff, L. . Serviço De Tradução De Linguagens De Marcação Para A Internet. Anais Do Simpósio Brasileiro De Redes De Computadores, 2007. V. Vol. 1. P. 541-554.

Perlin, C. B.; Santana, L. H. Z. ; Martins, D.S. ; Prado, A. F. ; Souza, W.L.; Biajiz, M. . Um Ambiente De Computação Ubíqua Para O Ensino Baseado Em Pbl Anais Da Xxxiv Conferencia Latinoamericana De Informática, 2007.

Santana, L. H. Z. ; Prado, A. F. ; Souza, W.L.; Biajiz, M. . Usando Ontologias, Serviços Web Semânticos E Agentes Móveis No Desenvolvimento Baseado Em Componentes. Simpósio Brasileiro De Componentes, Arquiteturas E Reutilização De Software, 2007. V. 1. P. 163-176.

Perlin, C. B. ; Prado, A. F. ; Santana, L. H. Z. . Framework Para Adaptação De Conteúdo Em Celulares. Anais da Jornada Científica Da Ufscar, 2007. V. 3. P. 35.

Santana, L. H. Z. ; Prado, A. F. . Um Modelo De Ontologia Para Adaptação De Conteúdo Da Internet. In: 7º Jornada Científica Da Ufscar, 2007, São Carlos. V. 3. P. 1300-1302.

Santana, L. H. Z. ; Martins, D.S. ; Perlin, C. B. ; Prado, A. F. ; Souza, W.L.; Biajiz, M. . Adaptação De Páginas Web Para Dispositivos Móveis. Do Simpósio Brasileiro De Sistemas Multimídia E Web. Gramado, 2007. V. 1. P. 1-8.

Perlin, C. B. ; Santana, L. H. Z. ; Prado, A. F. . Um Framework Para Adaptação De Conteúdo Em Celulares. Anais Do Workshop On Ongoing Undergraduate Research - Simpósio Brasileiro De Sistemas Multimídia E Web, 2007. P. 225-227.

Santana, L. H. Z. ; Prado, A. F. ; Souza, W.L.; Biajiz, M. . Uma Abordagem Para A Adaptação De Conteúdo Da Internet. Anais Do Ongoing Thesis And Dissertations. P. 206-210.

Santos, H. F. ; Santana, L. H. Z. ; Martins, D.S. ; Souza, Souza, W.L.; Prado, A. F. ; Biajiz, M. . An Ubiquitous Environment For Medical Education Anais do Symposium On Applied Computing, a ser publicado, 2008.

Martins, D.S. ; Santana, L. H. Z. ; Biajiz, M. ; Prado, A. F. ; Souza, W.L.. Context-Aware Information Retrieval On An Ubiquitous Medical Learning Environment Anais do Symposium On Applied Computing, a ser publicado, 2008.