EDUARDO FELIPE ZAMBOM SANTANA

Uma Abordagem Orientada por Modelos para o Desenvolvimento de Software na Computação Ubíqua

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

SÃO CARLOS
2010

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

S232ao

Santana, Eduardo Felipe Zambom.
Uma abordagem orientada por modelos para o desenvolvimento de software na computação ubíqua / Eduardo Felipe Zambom Santana. -- São Carlos : UFSCar, 2010.
92 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2010.

1. Engenharia de software. 2. Computação ubíqua. 3. Ontologia. 4. Desenvolvimento orientado por modelos. 5. Reuso. I. Título.

CDD: 005.1 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

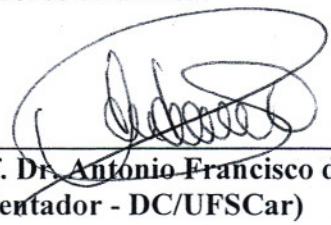
Programa de Pós-Graduação em Ciência da Computação

"Uma Abordagem Orientada por Modelos para o Desenvolvimento de Software na Computação Ubíqua"

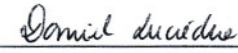
EDUARDO FELIPE ZAMBOM SANTANA

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. Daniel Lucrédio
(DC/UFSCar)

Profa. Dra. Itana Maria de Souza Gimenes
(Departamento de Informática /UEM)

São Carlos
julho/2010

Sonho que se Sonha só

É só um sonho que se sonha só,

Mas sonho que se sonha junto é realidade

Raul Seixas

Agradecimentos

Agradeço primeiramente a meus pais que sempre me apoiaram. A minha namorada Brianda que me ajudou bastante desde os tempos da graduação e a meus irmãos que diretamente e indiretamente sempre contribuiram em todos os aspectos de minha vida. Agradeço a meus amigos, principalmente os mais antigos, Enio, Marcelo, Sandro, Zé e Robinho e aos feitos em São Carlos, na graduação e no mestrado, especialmente Fausto, Rafael, Vitor, Raphael, Mateus, Alexandre e Bernardo.

Agradeço aos professores Prado e Wanderley pela orientação no trabalho desenvolvido e as Professoras Valéria, Sissi e especialmente a professora Volia na ajuda no desenvolvimento do PRE. Dedico também este trabalho ao professor Mauro Biajiz, que faz muita falta ao Departamento de Computação da UFSCar.

Lista de Figuras

Lista de Tabelas e Listagens

Lista de Abreviações

ABP	Aprendizagem Baseada em Problemas
ATL	ATLAS Transformation Language
AOSD	Apect-Oriented Software Development
API	Application Programming Interface
CASE	Computer Aided Software Engineering
CDC	Connected Device Configuration
CHTML	Compact HTML
DSM	Domain-Specific Modeling
EMF	Eclipse Modeling Framework
EMP	Eclipse Modeling Project
GCU	Grupo de computação ubíqua
GEF	Graphical Editing Framework
GMF	Graphical Modeling Framework
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JET	Java Emitter Template
JEE	Java Enterprise Edition
JME	Java Micro Edition
MDA	Model-Driven Architecture
MDD	Model-Driven Development
MVCASE	Multiple-View CASE
P2P	Peer-to-Peer
PBL	Problem-Based Learnig
PC	Personal Computer
PDA	Personal digital assistants
PDS	Processo de desenvolvimento de software
PRE	Portfólio Reflexivo Eletrônico
SDK	Software Development Kit
SQL	Structured Query Language
SO	Sistema Operacional
UML	Unified Modeling Language

WML	Wireless Markup Language
XML	Extended Markup Language
XMI	XML Meta-Data Interchange

Resumo

Com o avanço da capacidade de hardware e de tecnologias chaves de software e de redes, a Computação Ubíqua está se tornando uma realidade. O termo Computação Ubíqua refere-se a ambientes saturados de dispositivos computacionais e redes de comunicação, que se integram naturalmente à atividade humana. Segundo Mark Weiser, o pai da computação ubíqua, "as mais profundas tecnologias são as que desaparecem". Neste sentido a Computação Ubíqua pode ser considerada o oposto da Realidade Virtual. Enquanto na segunda o usuário penetra no mundo virtual, na primeira é a computação que penetra no mundo físico do usuário, construindo a ligação entre os dois mundos.

Pesquisas, em diferentes áreas, têm sido realizadas para melhorar processo de desenvolvimento de software na Computação Ubiqua. Na Engenharia de Software, este novo paradigma, principalmente devido à quantidade e diversidade de dispositivos e plataformas, apresenta problemas, como: desenvolvimento das aplicações de forma manual e sob demanda; e dificuldade de manutenção e evolução das aplicações para atender aos novos requisitos e acompanhar as mudanças de plataformas. Esses problemas têm motivado a pesquisa de métodos, processos, técnicas e ferramentas para modelagem, implementação, testes para apoiar o desenvolvimento de aplicações ubíquas.

Motivados em pesquisar uma solução para esses problemas, este projeto desenvolveu uma abordagem orientada por modelos para a construção de software na computação ubíqua. A abordagem baseia-se na Modelagem Específica de Domínio (Domain Specific Modeling – DSM) e na Arquitetura Orientada por Modelos (Model-Driven Architecture - MDA). Com foco na reutilização de software, em diferentes níveis do

ciclo de vida do software desde a modelagem da aplicação até sua implementação, a abordagem possibilita um ganho de produtividade no desenvolvimento de aplicações que devem ser executadas em diferentes arquiteturas da computação ubíqua, conforme os inúmeros dispositivos móveis, como celulares, PDAs e outros. Os principais mecanismos que automatizam parte das atividades do Engenheiro de Software na execução da abordagem são: uma ferramenta CASE, um gerador de código para a modelagem de aplicações específicas de um domínio, a IDE (Integrated Development Environment) Eclipse e um framework para ciência de contexto e adapatação de conteúdo. Estudos de casos no domínio da Educação Médica foram desenvolvidos para testar e avaliar a abordagem proposta.

Abstract

With the advancement of hardware capability and key technologies of software and networking, ubiquitous computing is becoming a reality. The term ubiquitous computing refers to environments saturated with computing devices and communication networks that integrate naturally to human activity. According to Mark Weiser, the father of ubiquitous computing, "the most profound technologies are those that disappear." In this sense, ubiquitous computing can be considered the opposite of virtual reality. While the second one the user enters the virtual world, the first computing is penetrating into the user's physical world, building the connection between the two worlds.

Research in different areas have been undertaken to improve the process of software development in ubiquitous computing. In software engineering, this new paradigm, mainly due to the amount and diversity of devices and platforms, present problems, such as applications development in manual and on-demand, and difficulties in maintenance and development of applications to meet the new requirements and monitor changes platforms. These problems have motivated the search for methods, processes, techniques and tools for modeling, implementing, testing to support the development of ubiquitous applications.

Motivated to search for a solution to these problems, this project developed a model-driven approach to building software in ubiquitous computing. The approach is based on Domain-Specific Modeling (DSM) and Model-Driven Architecture (MDA). With a focus on reuse of software at different levels in the life cycle of software from the application modeling through implementation, the approach provides a productivity gain in application development that must be performed on different architectures of ubiquitous

computing, as the numerous mobile devices such as cellphones, PDAs and others. The main mechanisms that automate part of the activities of a Software Engineer in implementing the approach are: a CASE tool, a code generator for the modeling of specific applications of a field, the IDE (Integrated Development Environment) Eclipse is a framework for science context and content adaption. Case studies in the field of medical education were developed to test and evaluate the proposed approach.

Sumário

1. Introdução

O termo Computação Ubíqua refere-se a ambientes saturados de dispositivos computacionais e redes de comunicação, que se integram naturalmente à atividade humana. Segundo [WEI94], "*as mais profundas tecnologias são as que desaparecem*". Neste sentido a Computação Ubíqua pode ser considerada o oposto da Realidade Virtual. Enquanto na segunda o usuário penetra no mundo virtual, na primeira é a computação que penetra no mundo físico do usuário, construindo a ligação entre os dois mundos.

Com o avanço da capacidade de hardware e de tecnologias chaves de software e de redes, a Computação Ubíqua está se tornando uma realidade [PHA07]. Entretanto o desenvolvimento de software para este novo paradigma ainda possui diversos problemas, como: desenvolvimento de forma manual e sob demanda; falta de capacidade de expansão; e dificuldade de manutenção e evolução [HEL05]. Tais dificuldades podem ser observadas pela falta de metodologias, processos, técnicas e ferramentas para modelagem, implementação, teste e manutenção [SPI07] de software na computação ubíqua.

1.1. Proposta

Com o objetivo de aumentar a produtividade e melhorar a qualidade no desenvolvimento de aplicações na computação ubíqua, através do reúso de software e do uso de ferramentas que auxiliam o projeto e a implementação de aplicações, esse trabalho propõe uma abordagem orientada por modelos para o desenvolvimento de software na computação ubíqua baseada na Modelagem Específica de Domínio (*Domain-Specific Modeling – DSM*) [KEL08] e na Arquitetura Orientada por Modelos (*Model-Driven Architecture – MDA*) [OMG08].

Esta abordagem parte da elaboração de um perfil UML que representa as principais abstrações encontradas em um domínio específico e possibilita a modelagem de diferentes aplicações desse domínio. Perfis que caracterizam as diferentes plataformas de implementação que poderão ser utilizadas nas aplicações ubíquas. A partir dos modelos que representam uma aplicação do domínio em uma plataforma específica, é possível gerar grande parte do código das aplicações [SAN09]. A modelagem e a geração de código das aplicações podem ser facilitadas com o suporte de uma ferramenta CASE e um gerador de código, que aumentam a produtividade e melhoram a qualidade das aplicações [HAR00].

Baseado na Modelagem Específica de Domínio, o processo de desenvolvimento de software proposto é dividido em duas etapas: Engenharia de Domínio (ED) e Engenharia de Aplicação (EA).

Na ED o domínio das aplicações que serão desenvolvidas é estudado, e a partir desse estudo é elaborado o perfil UML que representa as principais abstrações desse domínio. Também são elaborados os perfis das plataformas da computação ubiqua. Além disso, são construídos os apoios computacionais utilizados na Engenharia de Aplicação como: Integração dos Perfis em uma ferramenta CASE, geradores de código para diferentes plataformas de implementação e regras de transformações que automatizam as transformações entre modelos conforme as idéias da arquitetura orientada por modelos.

Ontologias [GUI05] são utilizadas para facilitar a descrição do domínio e a definição do seu perfil. Além disso, é possível inserir esse perfil em uma ferramenta CASE e construir um gerador de código que, juntos, facilitam as atividades do engenheiro de software na construção das aplicações do domínio. Para a implementação dos perfis, do gerador de código, e das transformações entre modelos, foi utilizado o Eclipse Modeling Project [EMP10].

Na EA são desenvolvidas as diferentes aplicações do domínio com o apoio dos recursos construídos na ED. Nessa etapa, são utilizados conceitos da arquitetura orientada por modelos, facilitando a transformação de modelos independentes de plataforma em modelos dependentes de plataforma e na geração do código fonte das aplicações a partir dos modelos dependentes de plataforma.

1.2. Contribuições

A principal contribuição desse trabalho é a abordagem sistemática para o desenvolvimento de software na Computação Ubíqua, que se destaca pelo emprego combinado de modelagem específica de domínio, arquitetura orientada por modelos, ontologias e o uso de uma ferramenta CASE, frameworks e um gerador de código.

Com essa abordagem foram observadas diversas melhorias no desenvolvimento de aplicações ubíquas, como: quantidade de artefatos reutilizados, qualidade das aplicações desenvolvidas, diminuição no tempo para o desenvolvimento e manutenção futura das aplicações.

Os estudos de caso desenvolvidos para avaliar a abordagem proposta e apresentados nessa dissertação contribuem para facilitar o entendimento e a transferência dos resultados produzidos nesta pesquisa. Este estudo consiste no desenvolvimento dos artefatos e ferramentas no domínio da educação médica, validado no curso de medicina da Universidade Federal de São Carlos e na implementação de três aplicações desse domínio.

1.3. Organização

Esta dissertação está organizada conforme descrito a seguir.

O Capítulo 2 apresenta uma visão geral sobre a Computação Ubíqua e propostas relevantes e relacionadas com esse trabalho para o desenvolvimento de software nesse novo paradigma computacional.

O Capítulo 3 aborda os conceitos e técnicas utilizadas para o desenvolvimento da abordagem proposta, entre esses conceitos destacam-se: Ontologias, utilizada para facilitar a descrição de domínios; Eclipse Modeling Project (EMP), projeto que oferece especificações, ferramentas e frameworks que facilitam a implementação de ferramentas que auxiliam no processo de desenvolvimento de software; Arquitetura Orientada por Modelos e Modelagem Específica de Domínio, abordagens utilizadas como base para a definição da abordagem de desenvolvimento de software proposta neste trabalho; a ferramenta MVCASE [SAN09], uma ferramenta CASE utilizada para a inserção dos perfis UML e a modelagem das aplicações do domínio; e o framework UBICK [SAN08], utilizado como base para a implementação de um framework de domínio, e que implementa requisitos não funcionais da computação ubíqua.

O Capítulo 4 apresenta a abordagem de desenvolvimento de software proposta nesse trabalho, apresentando as duas fases da abordagem, Engenharia de Domínio e Engenharia de Aplicação, detalhando cada uma das disciplinas e atividades desenvolvidas nas duas fases.

O Capítulo 5 apresenta um estudo de caso no domínio de aplicações ubíquas na educação médica, desenvolvido para avaliar a abordagem proposta neste trabalho, validado no curso de medicina da Universidade Federal de São Carlos. Nesse domínio foram implementadas três aplicações diferentes.

O Capítulo 6 apresenta os resultados obtidos de uma comparação do desenvolvimento de uma das aplicações do estudo de caso, utilizando primeiro a abordagem proposta, e depois uma abordagem genérica para o desenvolvimento de software.

Finalmente, o Capítulo 7 apresenta as conclusões da dissertação, apresentando os resultados, contribuições e limitações da abordagem, também são discutidas as dificuldades encontradas no desenvolvimento da abordagem e são apontadas possilidades de trabalhos futuros.

2. Computação Ubíqua

A Computação Ubíqua vem sendo considerada uma nova era da computação. A primeira era foi representada pelos *Mainframes*, que eram máquinas compartilhadas por uma grande quantidade de usuários. A segunda era é representada pelos Computadores Pessoais (Personal Computers – PC), onde cada computador é destinado a apenas um usuário. A terceira era, chamada também de "*Calm Technology*", tem como objetivo fazer com que os computadores presentes em um certo ambiente auxiliem os humanos nas suas tarefas sem que sejam notados [WEI91].

O termo Computação Ubíqua, que nomeia a terceira era foi cunhado por Mark Weiser enquanto pesquisador do XPARC (*Xerox Palo Alto Resarch Center*) no final da década de 80. A pesquisa inicial partiu da percepção de problemas de usabilidade dos computadores pessoais como: alta complexidade, alta necessidade de atenção por parte do usuário e o isolamento entre computadores, pessoas e suas atividades. Assim, o objetivo dos pesquisadores do XPARC era transportar a computação para segundo plano, permitindo uma maior interação entre humanos, por diminuir a necessidade de interação com computadores [WEI99].

Nos anos que se passaram desde a publicação do primeiro artigo sobre a Computação Ubíqua, foram identificadas várias áreas de interesse para pesquisas em torno desse novo paradigma. Dentre elas destacam-se [LYY02]:

a) *Computação Móvel*: o advento dos laptops e das Wirelles Local Networks (WLANS) no começo dos anos 1990 trouxe a necessidade do desenvolvimento de sistemas

distribuídos para clientes móveis, ou computação móvel. Esse campo da ciência da computação forma a base da computação ubíqua;

b) *Intenção do Usuário*: para que as aplicações ubíquas atinjam o objetivo de não serem notadas, se faz necessário que a intenção dos usuários seja capturada de forma implícita. De outra maneira, seria impossível determinar quais ações a aplicação deverá tomar em determinado momento;

c) *Adaptação*: uma estratégia de adaptação é necessária quando existe uma diferença significativa entre os recursos e as necessidades. Na computação ubíqua, os recursos em questão podem ser: largura de banda, memória, etc; e as necessidades podem ser: vídeo, áudio, notícias, etc.

d) *Ciência de Contexto*: pela necessidade de ser o menos intrusivo possível se faz necessário que a aplicação conheça a situação do usuário e do ambiente em que ele está atuando, e possa modificar seu comportamento sem que sejam necessários comandos explícitos;

e) *Privacidade e Segurança*: esses temas são problemas herdados da computação móvel, que em computação ubíqua possuem uma complexidade adicional. Isso porque as atividades para captura da intenção e do contexto do usuário produzem uma grande quantidade de informações privadas.

A evolução dessas áreas deverá possibilitar a criação de aplicações ubíquas como previsto por Mark Weiser. Alguns fatos como o aumento da utilização da Internet por dispositivos móveis, chegando a superar a utilização por Desktop em alguns paises [WIL06], o crescimento no número de celulares no Brasil [ANT07] e estudos que mostram que o usuário típico de uma cidade média americana mantém seu aparelho de celular ligado e próximo durante 85% por cento do tempo [PAT06], mostram que os diversos tipos de

computadores estarão presentes cada vez mais na vida das pessoas, cabendo aos pesquisadores buscarem soluções que facilitem seu uso.

A próxima seção apresenta uma revisão bibliográfica sobre abordagens para o desenvolvimento de software na computação ubíqua.

2.1. Engenharia de Software na Computação Ubíqua

Com o objetivo de correlacionar a proposta dessa dissertação, com os trabalhos que abordam o mesmo tema, essa seção apresenta uma revisão bibliográfica da literatura em relação à Engenharia de Software (ES) na Computação Ubíqua.

Nessa revisão foi feita a seguinte pergunta: quais são as técnicas da ES propostas para uso na Computação Ubíqua?

Para responder a esta pergunta, os critérios para seleção de fontes de consulta foram: disponibilidade de artigos via Web; existência de mecanismos de busca através de palavras-chave; e tratamento de pelo menos uma das características apresentadas da Computação Ubíqua. Para escolha das palavras-chave, 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*. Como critérios de inclusão foram usados: os artigos devem estar disponíveis nos mecanismos de busca *ACM Portal*, *IEEE Xplore*, *Elsevier* e *SpringerLink*; os artigos contêm textos completos e foram publicados no período de 2006 a 2009. Baseado nos resumos e as introduções dos artigos, estes foram selecionados 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 [BAL06]	Framework	Ciência de Contexto, Onipresença de Serviços, Capacidade de Adaptação, *Invisibility*
Soroker [SOR06]	Framework	Capacidade de Adaptação, Ciência de Contexto
Vale [VAL09]	Metodologia / Processo	Ciência de Contexto
Jung [JUN06]	Metodologia/ Processo	Interoperabilidade, Heterogeneidade, Capacidade de Adaptação
Carton [CAR07]	Metodologia/ Processo	Ciência de Contexto, Capacidade de Adaptação
Pham [PHA07]	Metodologia/ Processo	Ciência de Contexto, Capacidade de Adaptação
Fernandes [FER07]	Metodologia/ Processo	Capacidade de Adaptação, Ciência de Contexto
Bulcão Neto [BUL07]	Metodologia/ Processo	Ciência de Contexto
Burghardt [BUR08]	Ferramentas	Capacidade de Adaptação, Ciência de Contexto, Interoperabilidade
Riva [RIV06]	Padrões de Projeto	Ciência de Contexto, Interoperabilidade
Min [MIN07]	Modelo de Programação	Capacidade de Adaptação, Ciência de Contexto
Devdatta [KUL07]	Modelo de Programação	Ciência de Contexto

Os artigos selecionados apresentam diferentes propostas para o desenvolvimento de aplicações ubíquas, destacando-se: frameworks, 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 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; e aplicação de regras, que interpreta as modificações nos eventos.

O trabalho de Vale [VAL09] apresenta um processo orientado por modelos, chamado COMODE, para o desenvolvimento de aplicações cientes de contexto. Nesse trabalho é apresentado um metamodelo para a modelagem das aplicações cientes de contexto.

O trabalho de Jung [JUN06] apresenta um processo que combina a Model-Driven Architecture (MDA) e a *Agent-Oriented Software Development (AOSD*). Este trabalho argumenta que agentes de software facilitam a implementação de aplicações distribuídas autônomas, e que a MDA facilita o processo de desenvolvimento de software.

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 Burghardt [BUR08] apresenta um processo de desenvolvimento de software que disponibiliza uma série de ferramentas que auxiliam o desenvolvimento de aplicações ubíquas. Esse trabalho argumenta que essas ferramentas aumentam a produtividade e facilitam o desenvolvimento de software.

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 apresenta os padrões *Flexible Context Processing* e *Enactor*. O primeiro provê reúso de elementos do contexto, o segundo oferece controle para apoiar o 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.

2.2. Conclusões

De maneira geral, as abordagens têm foco na implementação parcial das aplicações ubíquas. Entretanto, são poucos os trabalhos que propõem abordagens que enfocam as diferentes fases do Processo de Desenvolvimento de Software (PDS) na Computação Ubíqua, desde os requisitos até a implementação e testes.

Nesse sentido, esta dissertação propõe uma abordagem de desenvolvimento baseada na Modelagem Específica de Domínio e na Arquitetura Orientada por Modelos, e que também emprega Ontologias, para facilitar a descrição de um domínio específico e que disponibiliza uma série de ferramentas e artefatos que facilitam a implementação das diferentes aplicações de um domínio.

Para facilitar o entendimento da abordagem proposta, o próximo capítulo apresenta uma visão geral sobre: Ontologias empregadas na descrição do domínio; Arquitetura Orientada por Modelos e Modelagem Específica de Domínio utilizadas na abordagem proposta; Eclipse Modeling Project utilizado para a implementação dos apoios computacionais; e o Framework UBICK, utilizado como base para a implementação de um framework de domínio.

3. Conceitos e Técnicas

Para facilitar o entendimento da abordagem proposta, esta seção apresenta os principais conceitos e técnicas utilizados no seu desenvolvimento, que incluem: Arquitetura Dirigida por Modelos [OMG08], Modelagem Específica de Dominio [KEL08], Eclipse Modeling Project [EMP10], MVCASE [SAN09B], Ontologias [GUI05] e o framework UBICK [SAN08].

3.1. Arquitetura Dirigida por Modelos

A Arquitetura Dirigida por Modelos (Model-Driven Architecture – MDA) é um conjunto de padrões que visa aumentar o nível de abstração em processos para o desenvolvimento de software, nessa abordagem os modelos são os principais artefatos do processo e estão presentes em todas as fases do desenvolvimento das aplicações, desde o levantamento de requisitos até a manutenção. Esses modelos devem ser definidos formalmente para que possam ser processados automaticamente por máquina. Entre esses modelos destacam-se:

a) Computational Independent Model (CIM): O CIM é o modelo de mais alto nível de abstração, e representa apenas os requisitos de um domínio ou aplicação, sem nenhum detalhe computacional. Para esse nível pode ser usado, por exemplo, uma ontologia que descreve os principais conceitos do domínio ou de uma aplicação.

b) Platform-Independent Model (PIM): O PIM também é um modelo também com alto nível de abstração, mas já são considerados alguns componentes computacionais da aplicação. Para esse nível podem ser utilizados diagramas da UML, sem detalhes da plataforma de implementação utilizada.

c) Platform-Specific Model (PSM): O PSM, é um modelo bem próximo da implementação final da aplicação. Ele representa, além dos requisitos da aplicação, também componentes específicos da plataforma, da arquitetura e de padrões de projeto utilizados para a implementação de uma aplicação. Para esse nível podem ser utilizados diagramas completos da UML com perfis específicos para uma plataforma.

Outro conceito fundamental na MDA são as transformações [OMG08], a MDA define dois tipos de transformações, as transformações entre modelos (Model-to-Model - M2M) e as transformações de modelo em código (Model-to-Text - M2T). A primeira (M2M) representa as transformações entre modelos e definem regras para a transformação automática de um CIM em um PIM ou de um PIM em um PSM. A segunda (M2T) define as regras de transformações do PSM para código fonte.

3.2. Modelagem Específica de Domínio

Em conjunto com a MDA, que propõe que grande parte do desenvolvimento de software seja feita através de modelos e transformações, é possível combinar também a Modelagem Específica de Domínio (Domain-Specific Modeling - DSM) [KEL08] para a definição do processo de desenvolvimento de software.

Processos baseados na DSM descrevem os conceitos e terminologias específicas de um determinado domínio [KEL08], em um nível de abstração que possibilita a definição de modelos que ofereçam suporte automatizado para o projeto e a implementação de diferentes aplicações desse domínio.

A DSM tem dois objetivos principais [LAF08]:

a) *Aumentar o nível de abstração no processo de desenvolvimento de software*: Facilitar a implementação das aplicações de um determinado domínio, especificando uma solução em um alto nível de abstração.

b) *Gerar programas executáveis a partir dessas especificações*: Gerar programas diretamente dos modelos baseados nas especificações do domínio. Normalmente o código gerado é complementado por um framework específico do domínio que implementa funcionalidades comuns às aplicações do domínio.

A DSM define três artefatos principais em um processo, que são:

a) Descrição do Domínio: Um artefato que define o domínio considerado, para isso pode ser usado, por exemplo, um metamodelo específico do domínio, ou perfis UML. Esse artefato deve representar os principais conceitos, regras e abstrações do domínio, em um nível que permita a modelagem de diversas aplicações desse domínio.

b) Gerador de Código: Um gerador que a partir de um modelo específico do domínio gera grande parte do código fonte das aplicações do domínio para uma plataforma específica.

c) Framework de Domínio: Um framework que implementa requisitos do domínio que estão presentes em todas as aplicações do domínio, e também deve ser construído para uma plataforma específica.

3.3. Eclipse Modeling Project

O Eclipse Modeling Project [EMP10] é um projeto que disponibiliza diversas especificações e frameworks para facilitar a construção de ferramentas que apóiam processos baseados na MDA e na DSM. Esses frameworks facilitam a implementação de ferramentas CASE, geradores de código, regras de transformação entre modelos, e todos esses apoios são integrados na IDE Eclipse .

Nesse trabalho foram utilizados os seguintes frameworks: Java Emitter Templates (JET) e ATLAS Transformation Language (ATL) [ATL10].

Para a transformação de modelos em código (M2T), pode ser utilizado o framework Java Emitter Templates (JET). Com esse framework são definidos templates para uma plataforma alvo, que pode ser uma linguagem de programação, uma linguagem de marcação ou até um arquivo texto. Elementos do modelo são usados como parâmetros para os templates, sendo então gerado código fonte correspondente.

Para a implementação das regras de transformação entre os modelos (M2M), pode ser utilizado o framework Eclipse ATL. Nesse framework é possível definir regras de transformação que mapeiam elementos de um modelo para elementos de outro modelo.

Além disso foi utilizado um editor de perfis UML que depois de desenvolvidos podem ser integrados a ferramentas CASE que foram desenvolvidas com os frameworks do Eclipse Modeling Project, como por exemplo, a MVCASE.

3.4. MVCASE

A ferramenta *Multiple-View CASE (MVCASE)* , vem sendo desenvolvida desde 1997 por alunos de mestrado e iniciação científica do Departamento de Computação da UFSCar. Na sua ultima versão a MVCASE foi reconstruída para suportar a versão 2.1 da UML. Na sua implementação foram reutilizados os frameworks do Eclipse Modeling Project que facilitam a construção de ferramentas de modelagem. A MVCASE também permite a geração de código e a integração de frameworks, tanto nos modelos quanto na geração de código. A Figura 3.1 ilustra uma tela da ferramenta MVCASE, com um digrama de classes.

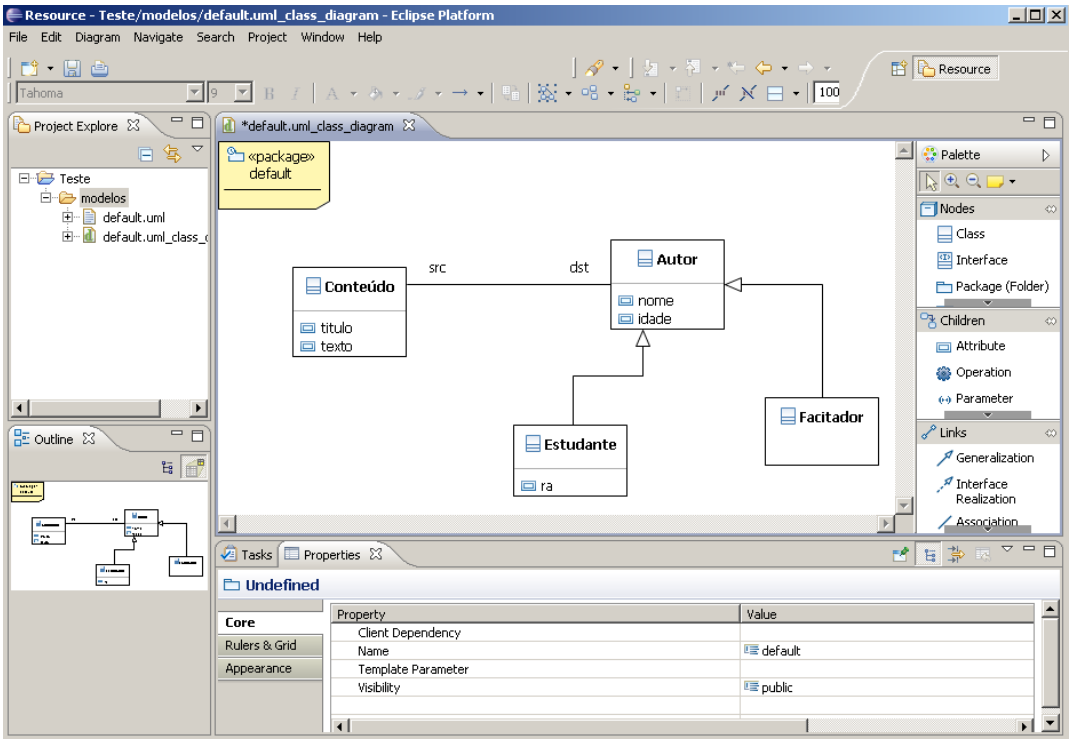


Figura 3.1. Diagrama de classe na ferramenta MVCASE.

3.5. Ontologias

Para a 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:

a) 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);

b) Indivíduos ou instâncias, componentes básicos de uma Ontologia, podendo incluir objetos concretos (e.g., uma pessoa) e abstratos (e.g., uma palavra);

c) Atributos, propriedades que caracterizam um objeto;

d) Relações são atributos que descrevem como os objetos se relacionam.

Dentre as várias linguagens propostas para 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.

Muitas pesquisas vêm utilizando Ontologias na Computação Ubíqua. Grande parte dessas pesquisas utiliza as Ontologias para descrição de contexto [BUL07]. 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.

3.6. Framework UBICK

O *Ubiquitous Computing Framework* (UCF) [SAN08] é um framework desenvolvido no Departamento de Computação da UFSCar e que implementa alguns requisitos não funcionais da computação ubíqua, entre eles:

(a) Ciência de Contexto, para possibilitar o uso das informações contextuais relativas ao ambiente computacional, pessoal e físico, na oferta de conteúdos e serviços personalizados;

(b) Adaptação de Conteúdo, para suportar o acesso ubíquo às aplicações, ou seja, para que essas aplicações possam ser acessadas de qualquer lugar, a qualquer hora e usando qualquer tipo de dispositivo;

(c) Comunicação Cliente/Servidor, o que permite que os dispositivos móveis conectem-se a servidores ou que os dispositivos conectem-se uns aos outros; e

(d) Descoberta, Onipresença e Composição de Serviços Web Semânticos, para permitir às aplicações o uso pleno dos serviços web disponíveis.

Neste trabalho o UBICK foi estendido para tornar possível também a comunicação P2P [SAN09D], o que permite que um dispositivo comunique-se diretamente com outro dispositivo.

3.7. Conclusões

Esta seção apresentou os conceitos e técnicas utilizadas no desenvolvimento da abordagem proposta.

A MDA foi utlizada como base para a abordagem proposta. Com isso o desenvolvimento de software começa em um alto nível de abstração, através de modelos independentes de plataforma. Com transformações definidas formalmente, esses modelos são transformados em modelos dependentes de plataforma e por fim é gerado o código fonte de uma aplicação. Com a MDA espera-se obter maior facilidade de manutenção nas aplicações desenvolvidas e maior grau de reúso.

A DSM foi combinado com a MDA para facilitar o processo de desenvolvimento de software. Na abordagem proposta é utilizada uma linguagem especifica para o domínio considerado para a modelagem das aplicações e a partir desses modelos podem ser construídos geradores de código para diferentes plataformas. Com a DSM espera-se obter maior facilidade na modelagem das aplicações de um domínio específico e uma maior produtividade, com a geração de grande parte do código fonte das aplicações do domínio.

Os frameworks do Eclipse Modeling Project foram utilizados para facilitar a implementação das ferramentas específicas do domínio, como os perfis de domínio e plataformas e as regras de transformação que são desenvolvidas para facilitar o desenvolvimento das diferentes aplicações do domínio. A Ferramenta MVCASE foi utilizada para a criação dos modelos específicos do domínio e para a geração de código.

Ontologias são utilizadas para facilitar a descrição de um domínio e para servir como base para o desenvolvimento dos perfis que representam as abstrações do domínio e possibilitam a modelagem das aplicações.

O framework UBICK foi utilizado como base para a construção do Framework do Dominío, pois ele já implementa requisitos não funcionais da computaçãso ubíqua que estão presentes na maioira das aplicações ubíquas.

4. Abordagem Orientada por Modelos para o Desenvolvimento de Software na Computação Ubíqua

Considerando as idéias apresentadas sobre a computação ubíqua, os estudos realizados na revisão bibliográfica, e as experiências do Grupo de Computação Ubíqua (GCU) da UFSCar, essa dissertação apresenta os resultados da pesquisa de uma Abordagem Orientada por Modelos para o Desenvolvimento de Software na Computação Ubíqua. A abordagem é baseada na Modelagem Específica de Domínio e na Arquitetura Orientada por Modelos. A criação de linguagens específicas de domínio é guiada por uma Ontologia que representa um domínio. A ferramenta MVCASE oferece o suporte computacional, em conjunto com o framework UBICK, um Gerador de Código e um Gerador de Testes Automatizados, construídos especificamente para cada domínio.

Conforme ilustra a Figura 4.1, baseado nas idéias de reúso de software, na MDA e na DSM o processo de desenvolvimento de software proposto nesse trabalho é realizado em duas grandes etapas: Engenharia de Domínio (ED) e Engenharia de Aplicação (EA). Essas duas etapas são interativas e incrementais, geram diferentes artefatos e são compostas por disciplinas com atividades, técnicas, e papéis associados.

Na ED, inicialmente o domínio do problema é descrito em uma ontologia, usada para representar o conhecimento do domínio. Baseado na ontologia, são construidos perfis UML, que permitem a modelagem de diferentes aplicações de um domínio específico. Esses perfis são de dois tipos: Perfil de Domínio e Perfis Dependentes de Plataforma. O perfil de domínio é usado para modelar as aplicações independentemente de plataforma,

produzindo os Modelos Independentes de Plataforma. Os perfis de plataforma consideram a plataforma adotada para o projeto e implementação da aplicação e possibiltam a transformação dos modelos independentes de plataforma em Modelos Dependentes de Plataforma.

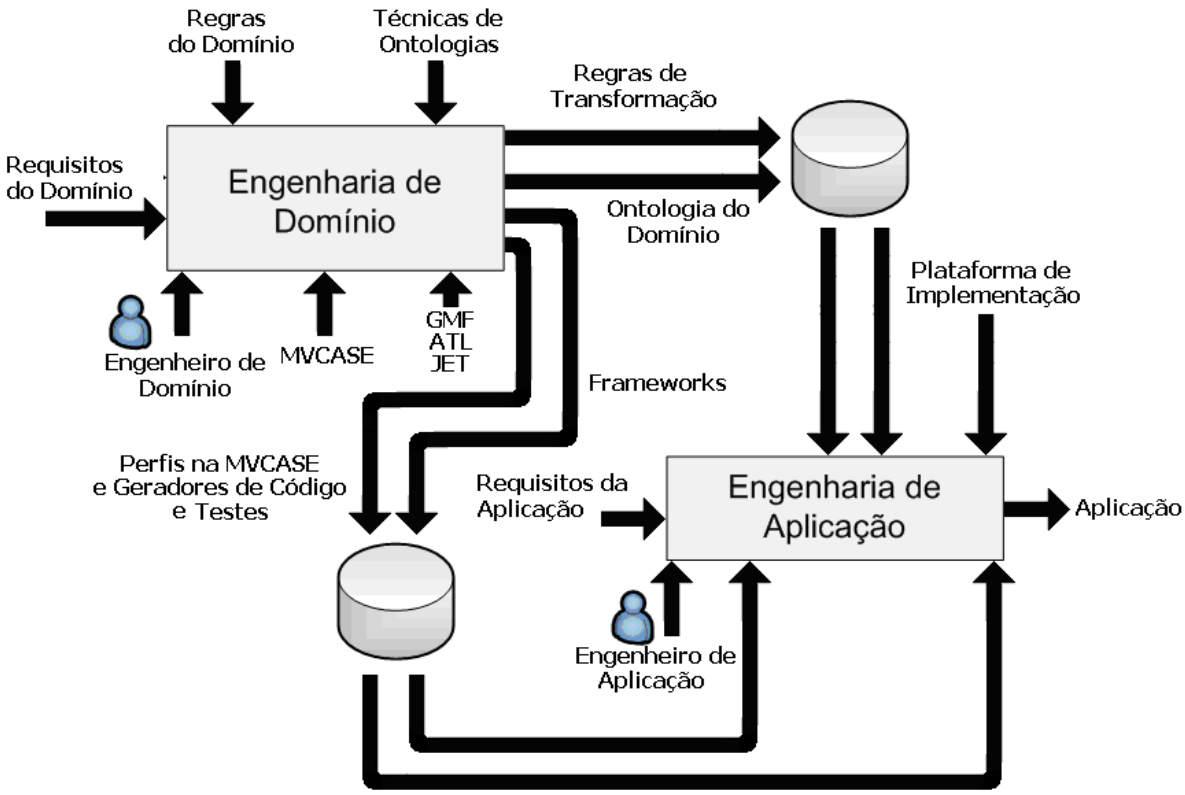


Figura 4.1. Processo de Desenvolvimento de Software.

Os perfis UML são integrados na ferramenta MVCASE facilitando a reutilização do conhecimento geral do domínio. Outro artefato produzido nessa etapa são Regras de Transformação, utilizadas para transformar os modelos independetes de plataforma em modelos dependentes de plataforma. Um gerador de código é implementado para automatizar grande parte do desenvolvimento das diferentes aplicações do domínio. Também é construído um gerador de testes para melhorar a qualidade das aplicações. Procurando tirar maior proveito no reúso, nessa etapa são ainda identificados frameworks

que possam apoiar o desenvolvimento das aplicações. Nessa atividade podem-se adotar frameworks já disponíveis no mercado que se adéquam ao processo de desenvolvimento.

Na EA o Engenheiro de Software, baseado nas idéias de reúso dos artefatos produzidos na ED, parte dos requisitos da aplicação, e com apoio da ontologia do domínio, dos perfis integrados na MVCASE, dos Geradores de Codigo e Testes e dos Frameworks definidos e usando as regras de transformação, desenvolve de forma mais rápida as aplicações do domínio.

4.1. Engenharia de Domínio

Conforme ilustra a Figura 4.2, a Engenharia de Domínio é realizada em três disciplinas: Análise do Domínio, Projeto do Domínio e Implementação do Domínio. Na Análise do Domínio são identificados, elicitados e modelados os requisitos do domínio do problema. Com base nos modelos de análise, é construída uma ontologia que descreve o domínio.

No Projeto do Domínio, inicialmente são tomadas decisões sobre a plataforma e arquitetura das aplicações. A partir da ontologia, obtida na Análise, é feita a modelagem de projeto do domínio com diagramas UML e é projetado um perfil que representa o domínio e um ou mais perfis que descrevem as possíveis plataformas de implementação das aplicações do domínio. Também são especificadas as variabilidades dos requisitos do domínio e são escolhidos os frameworks que poderão ser reutilizados no desenvolvimento das aplicações.

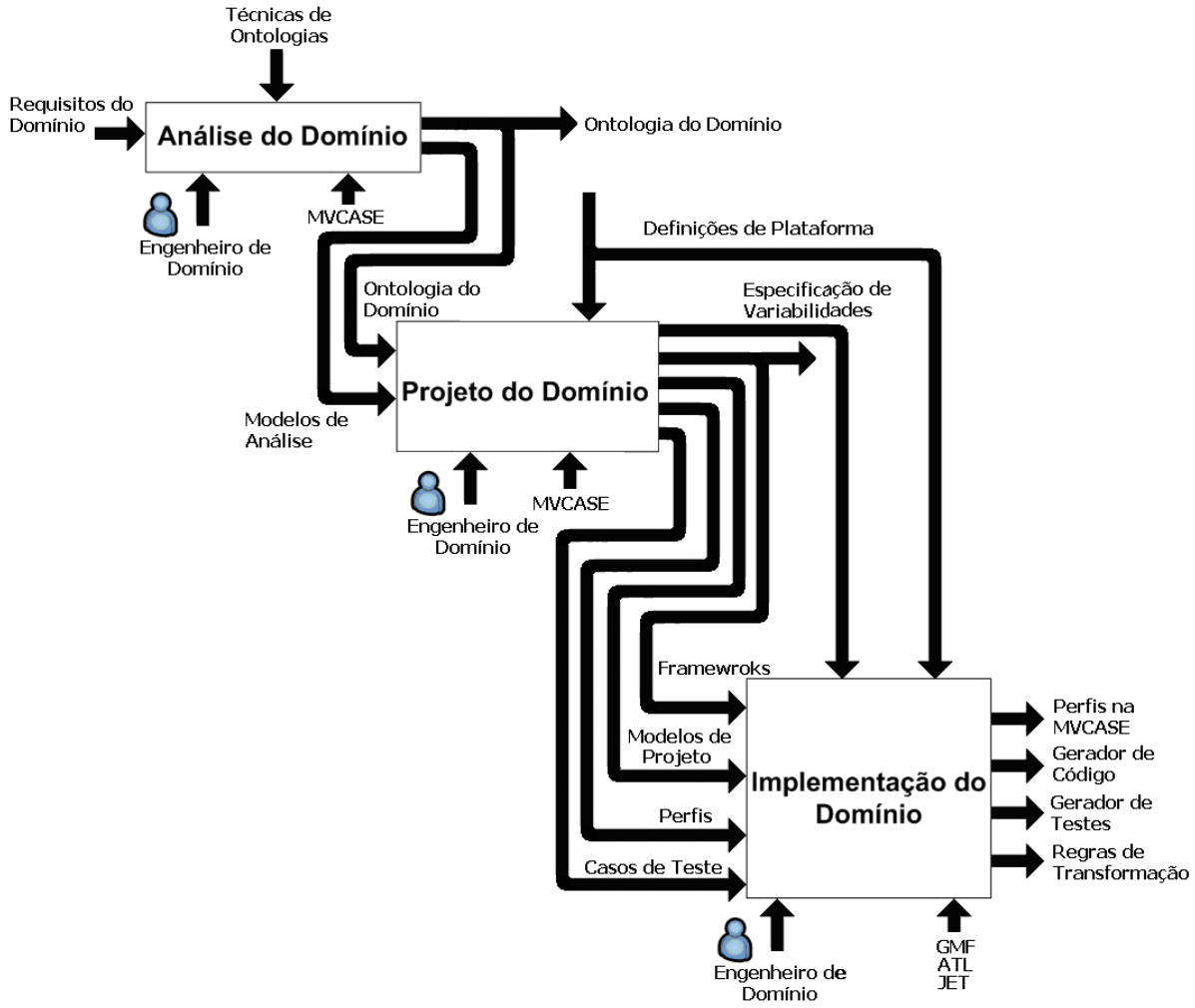


Figura 4.2. Engenharia de Domínio.

Na implementação do domínio, os perfis do domínio são integrados na ferramenta MVCASE, para suportar a modelagem das diversas aplicações do domínio. São implementados os geradores de código e de testes para automatizar parte da implementação das aplicações do domínio para uma plataforma específica. Também são descritas as regras de transformação dos modelos independentes de plataforma para os modelos dependentes de plataforma.

Para facilitar a evolução e a manutenção dos artefatos e ferramentas desenvolvidas na Engenharia de Domínio os seguintes padrões foram adotados:

a) Modelos, como a Ontologia e Diagramas UML, devem ser persistidos em arquivos que permitem controle de versão, como por exemplo, o XML Metadata Interchange (XMI) [GRO01].

b) Atualização dos artefatos: A Ontologia e os Perfis são atualizados, para acompanhar as decisões de projeto, e facilitar a manutenção futura.

c) Controle de versão de código, modelos e documentação: Todos os códigos fontes e documentações dos artefatos criados têm controle de versão, para facilitar o desenvolvimento.

Nas próximas seções são apresentadas as disciplinas da fase de Engenharia de Domínio, destacando seus fluxos de trabalho e os artefatos gerados. O domínio de Entrega de Cargas é usado para ilustrar os artefatos desenvolvimentos. Esse domínio representa a cadeia de entregas de uma empresa, onde devem ser cadastrados os clientes e as entregas diárias de uma empresa, e um software, com um algoritmo de roteirização, deve determinar a melhor rota para a realização dessas entregas.

4.1.1. Análise do Domínio

A Figura 4.3 ilustra o fluxo de atividades da disciplina análise do domínio. Primeiro é verificado se é um novo domínio a ser analisado ou se é uma modificação em um domínio já analisado. Se for um novo domínio, em Analisar Requisitos os requisitos do domínio são identificados, elicitados e descritos em um documento de requisitos. Se for uma mudança no domínio, em Analisar Novos Requisitos são analisados os novos requisitos do domínio, e esses requisitos são adicionados ao documento de requisitos.

O documento de requisitos é descrito textualmente com o título do requisito, uma descrição, um identificador único para cada requisito e sua prioridade. A Figura 4.4, por exemplo, ilustra parte de um documento de requisitos do domínio de entrega de cargas.

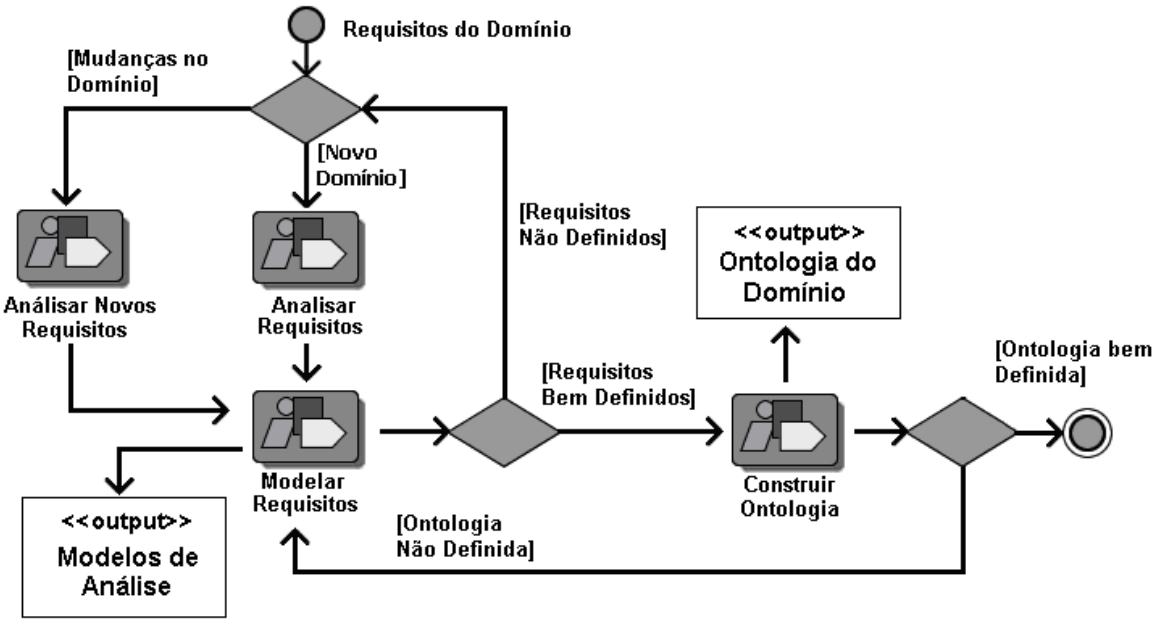


Figura 4.3. Fluxo de Atividades da Análise de Domínio.

[RF001] Cadastro de Veículos

Descrição do caso de uso: Deve ser possível cadastrar todos os veículos da frota das empresas, os campos que descrevem os veículos são: identificador, que identifica cada um dos veículos, placa, que contém a placa do veículo, motorista, nome do motorista do veículo e tipo do veículo, que descreve a categoria do veículo.

Prioridade: ☐ Essencial ■ Importante ☐ Desejável

[RF002] Cadastro de Produtos

Descrição do caso de uso: Deve ser possível cadastrar todos os produtos distribuidos ou coletados pelas empresas, os campos que descrevem os produtos são: identificador, que identifica cada um dos produtos, nome, que contém o nome do produto, peso, o peso de uma unidade do produto e valor, que descreve o valor do produto.

Prioridade: ■ Essencial ☐ Importante ☐ Desejável

[RF003] Cadastro de Clientes

Descrição do caso de uso: Deve ser possível cadastrar todos os clientes das empresas, os campos que descrevem os clientes são: identificador, que identifica cada um dos clientes, nome, que contém o nome do cliente, endereço, o endereço do cliente e telefone, que contém o telefone do cliente.

Prioridade: ■ Essencial ☐ Importante ☐ Desejável

Figura 4.4. Documento de Requisitos do Domínio de Entrega de Cargas.

Em seguida, na atividade Modelar Requisitos, os requisitos são modelados em diagramas de casos de uso e de tipos. Essas atividades se repetem até que a descrição do

domínio esteja bem definida. A Figura 4.5 ilustra um caso de uso do domínio de entrega de cargas e a Figura 4.6 ilustra parte do diagrama de tipos do domínio.

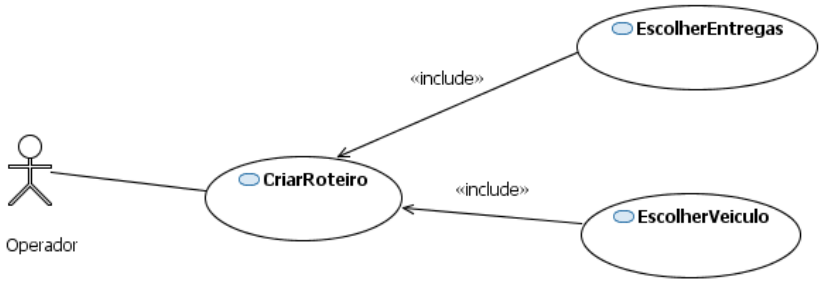


Figura 4.5. Casos de Uso do Domínio de Entrega de Cargas.

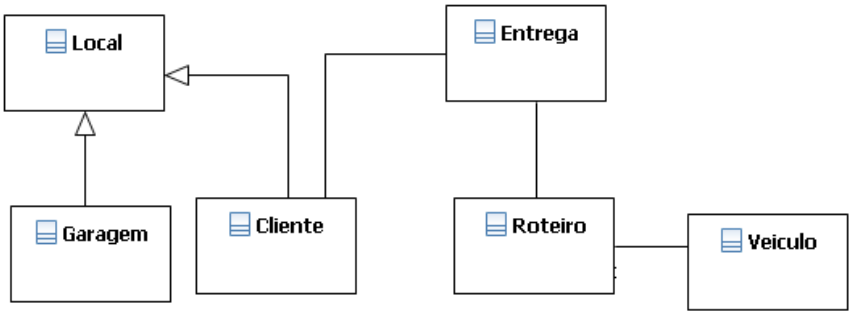


Figura 4.6. Diagrama de Tipos do Domínio de Entrega de Cargas.

Na atividade Construir Ontologia, é construida uma ontologia que representa o domínio. O objetivo dessa ontologia é facilitar o projeto do domínio. A Figura 4.7 ilustra como representar a ontologia desse domínio.

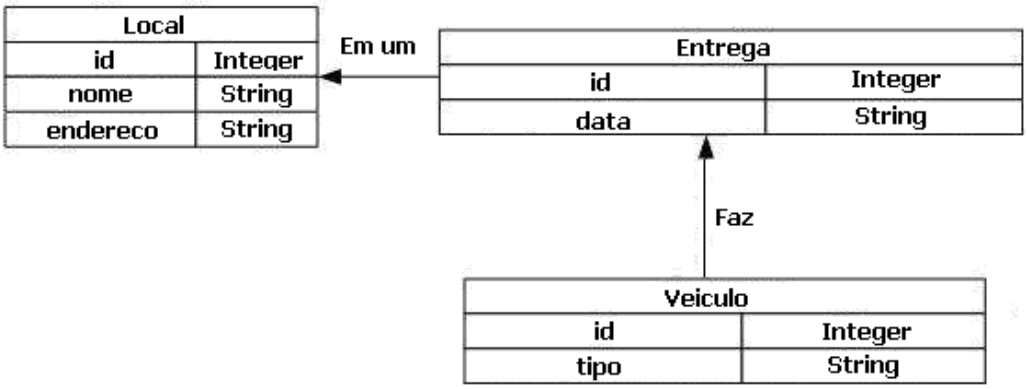


Figura 4.7. Ontologia do Domínio de Entrega de Cargas.

4.1.2. Projeto do Domínio

A Figura 4.8 ilustra o fluxo da disciplina de projeto do domínio. Em Definir Plataforma são definidas as possíveis plataformas das aplicações que serão desenvolvidas, como por exemplo, linguagem de programação, arquitetura, sistema gerenciador de banco de dados, protocolos, padrões de projeto entre outros. Isso é feito para que se possa construir os perfis das possíveis plataformas de implementação das aplicações e os geradores de código. As decisões de plataforma são armazenadas em um documento que registra essas decisões. A Figura 4.9 ilustra um exemplo desse documento para o domínio de entrega de cargas.

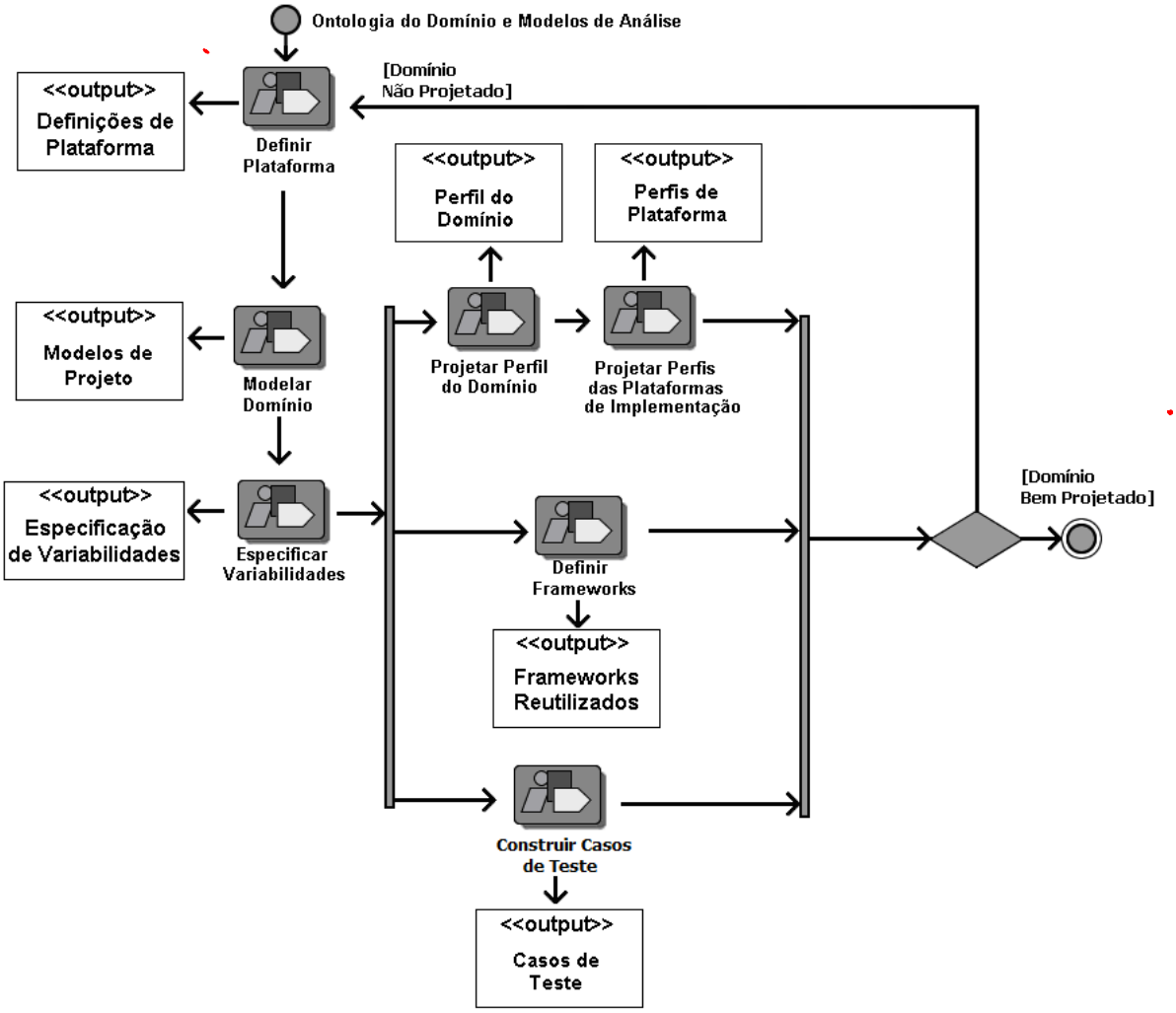


Figura 4.8. Fluxo de Atividades do Projeto do Domínio.

Elemento	Plataform Utilizada	Versão
SGBD	PostgreSQL	8.3
Linguagem de Programação	Java	6.0
Plataformas	JEE e JME	5.0 e 2.0
SO	Linux e Windows	X

Figura 4.9. Documento de Definição de Plataforma.

Na atividade Modelar Domínio, os modelos de análise são refinidos, considerando também elementos da plataforma escolhida, utilizando diagramas da UML, como o de classes e sequência. A Figura 4.10 ilustra um diagrama de classes de projeto do domínio de entrega de cargas.

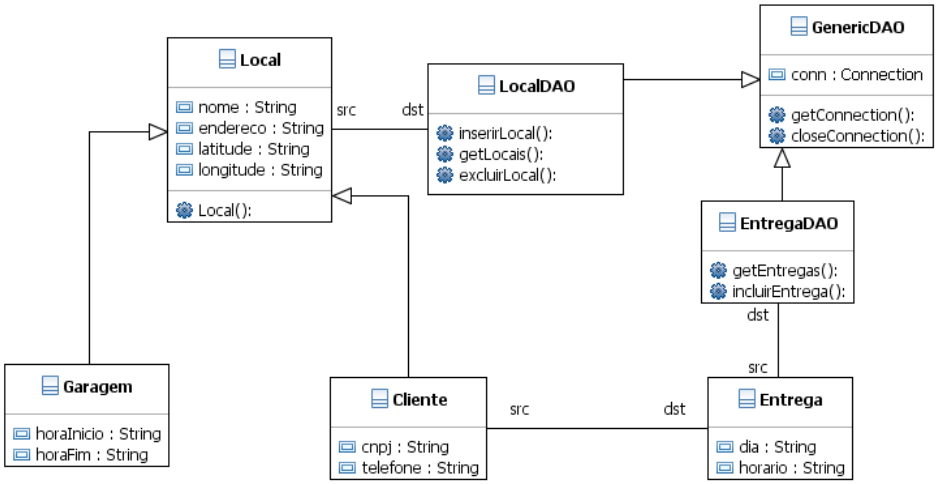


Figura 4.10. Diagrama de Classes do Domínio de Entrega de Cargas.

Depois que o domínio é modelado, são verificados os pontos variáveis dos requisitos do domínio, na atividade Especificar Variabilidades. Com isso é decidido quais os requisitos do domínio tem variações, e como implementá-las. A Figura 4.11 apresenta parte do documento que descreve as variabilidades do domínio.

Requisito	Descrição
RF001	Em algumas aplicações será necessário o campo Tipo Veículo.
RF016	O veículo poderá ou não, fazer mais de uma rota por dia.
RF022	Cada aplicação poderá escolher o módulo de roteirização utilizado.

Figura 4.11. Especificação das Variabilidades.

Depois, na atividade Definir Frameworks, são tomadas as decisões de quais frameworks podem ser utilizados no desenvolvimento das aplicações do domínio. A Figura 4.12 ilustra um documento que define os frameworks que poderão ser utilizados no desenvolvimento das aplicações do domínio.

Framework	Versão	Funcão
UBICK	1.0	Ciência de Contexto e Adaptação de Conteúdo em Dispositivos Móveis
Hibernate	3.3.2	Persistência de Dados
Java Server Faces	1.2	Implementação do Padrão MVC
Jasper Reports	3.7	Geração de Relatórios
Rich Faces	3.3	Componentes de Visão

Figura 4.12. Frameworks Utilizados no Desenvolvimento das Aplicações.

Na atividade Projetar Perfil do Domínio, a partir dos diagramas criados no projeto do domínio é definido o perfil para a modelagem das aplicações do domínio. A Figura 4.13 ilustra a definição do perfil para o domínio de entrega de cargas.

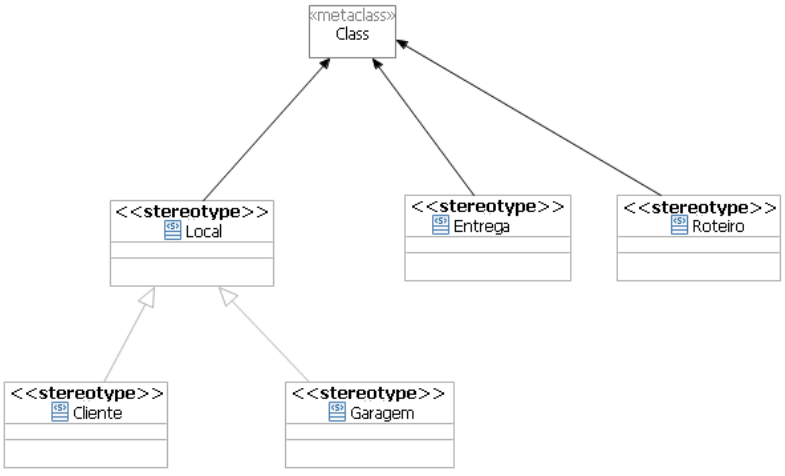


Figura 4.13. Descrição do Perfil do Domínio de Entrega de Cargas.

Além do perfil do domínio, na atividade Projetar Perfis de Plataforma, são definidos os perfis para as possíveis plataformas de implementação das aplicações do domínio. A Figura 4.14 ilustra, por exemplo, a definição do perfil para a plataforma JEE.

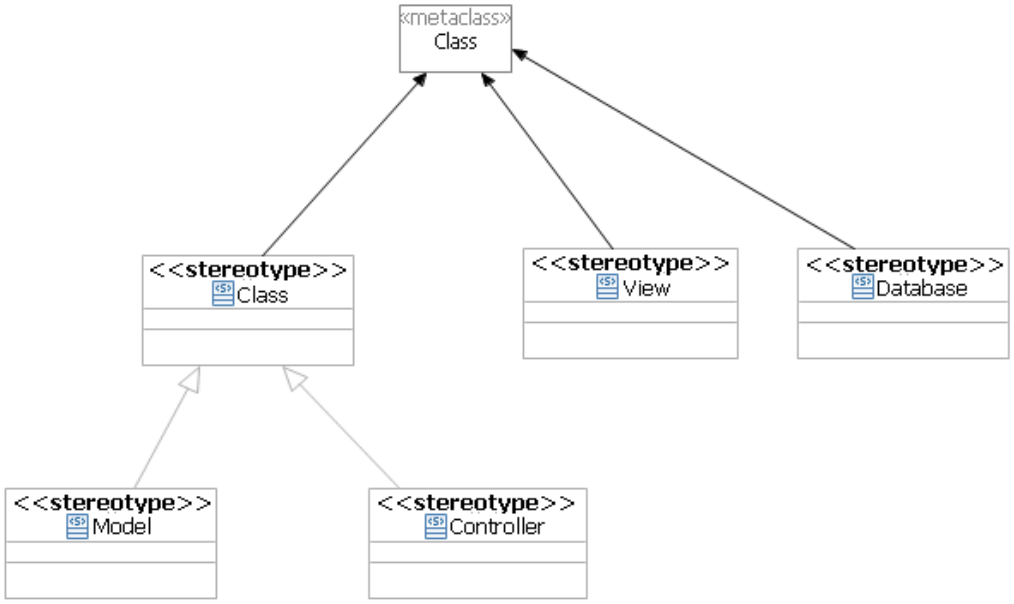


Figura 4.14. Descição do Perfil para a Plataforma JEE.

Na atividade Construir Casos de Testes, são escritos os casos de teste para os requisitos do domínio, esses casos de testes são utilizados posteriormente para a implementação do gerador de testes automatizados.

4.1.3. Implementação do Domínio

A Figura 4.15 ilustra o fluxo de atividades da disciplina de implementação do domínio. Na atividade Integrar Perfis na MVCASE, os perfis desenvolvidos para a modelagem das aplicações do domínio e para a modelagem das aplicações em uma plataforma específica são integrados na ferramenta MVCASE. A Figura 4.16 ilustra a ferramenta MVCASE com o perfil do domínio de entrega de cargas.

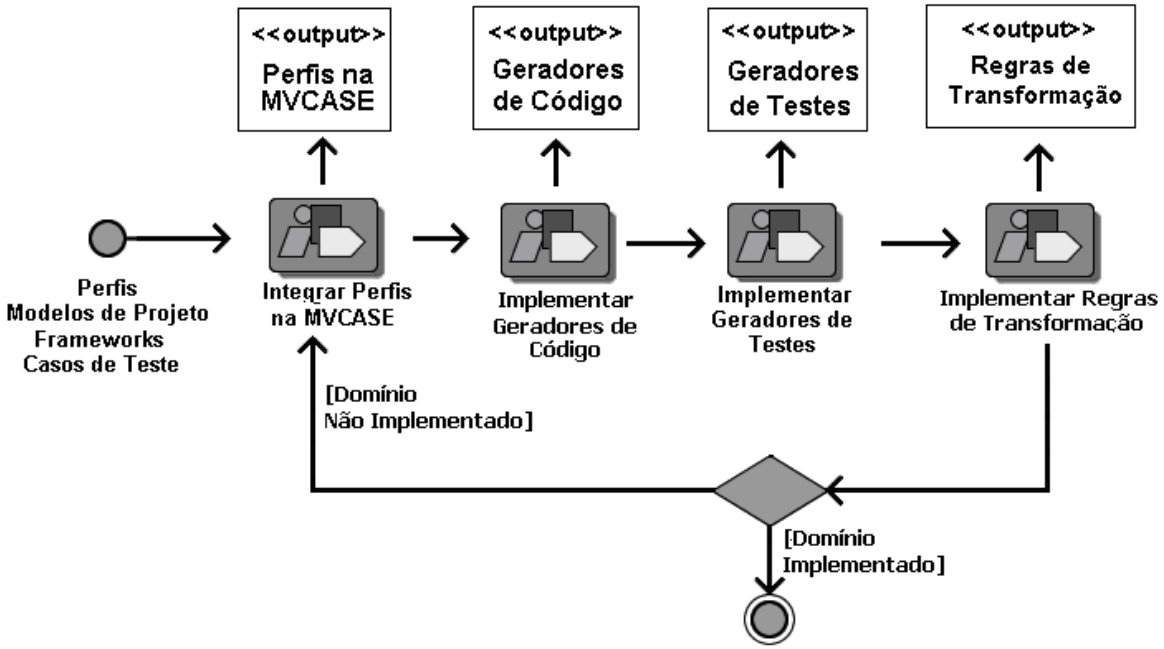


Figura 4.15. Fluxo de Atividades da Implementação do Domínio.

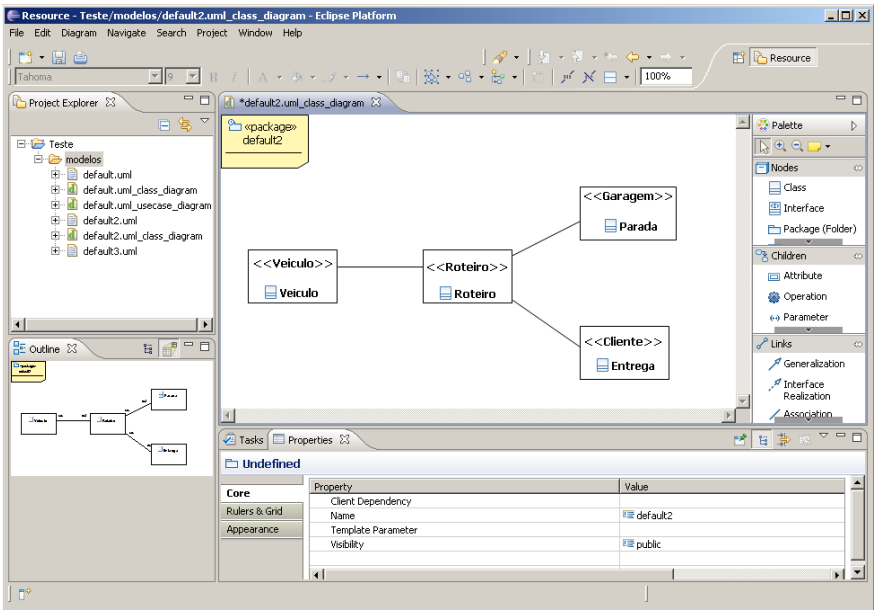


Figura 4.16. Diagrama de Classes do Domínio de Entrega de Cargas na MVCASE.

Na atividade Implementar Geradores de Código, é construído o gerador de código para a plataforma e a linguagem de programação definida no projeto do domínio, podem ser definidas geradores para diferentes plataformas e linguagens. Na atividade Implementar

Gerador de Testes são implementados os geradores de testes. A Figura 4.17 ilustra parte da descrição dogerador de código no framework JET para o domínio de entrega de cargas.

```
<%@taglib prefix="ws" id="org.eclipse.jet.workspaceTags" %>

<ws:file template="templates/dao/GenericDAO.jet" path="{$org.eclipse.jet.resou
<ws:file template="templates/controller/GenericController.jet" path="{$org.ecl


<ws:file template="templates/beans/Veiculo.jet" path="{$org.eclipse.jet.resour
<ws:file template="templates/dao/VeiculoDAO.jet" path="{$org.eclipse.jet.resou
<ws:file template="templates/controller/veiculoController.jet" path="{$org.ecl

<ws:file template="templates/beans/Entrega.jet" path="{$org.eclipse.jet.resour
<ws:file template="templates/dao/EntregaDAO.jet" path="{$org.eclipse.jet.resou
<ws:file template="templates/controller/EntregaController.jet" path="{$org.ecl
```

Figura 4.17. Gerador de Código do Domínio de Entrega de Cargas.

Em Implementar Regras de Transformação, são descritas as regras de transformação do modelos independentes de plataforma para os modelos dependentes de plataforma. A Figura 4.18 ilustra as regras de transformação do domínio de entrega de cargas.

```
rule Model2Model {

    from s:pim!Model
    to t:psm!Diagram(
        ownedElement <-s.ownedElement
    )

}
```

```
rule Veiculo2Model {

    from s:pim!Veiculo
    to t:psm!Model(
        ownedElement <- Sequence{o,p}
    ),
    o:psm!Controller(
    ),
    p:psm!Model(
    )

}
```

Figura 4.18. Regras de Transformação do Domínio de Entrega de Cargas.

4.2. Engenharia de Aplicação

Na fase de Engenharia de Aplicação da abordagem proposta, seguindo as idéias da Prototipagem Evolutiva [PRE06], o desenvolvimento da aplicação evolui dos requisitos em níveis mais abstratos até uma aplicação implementada e testada disponível para execução. Conforme ilustra a Figura 4.19, em cada fase do ciclo de vida da aplicação são aplicadas as

disciplinas conhecidas da Engenharia de Software: Análise, Projeto, Implementação, Testes e Implantação.

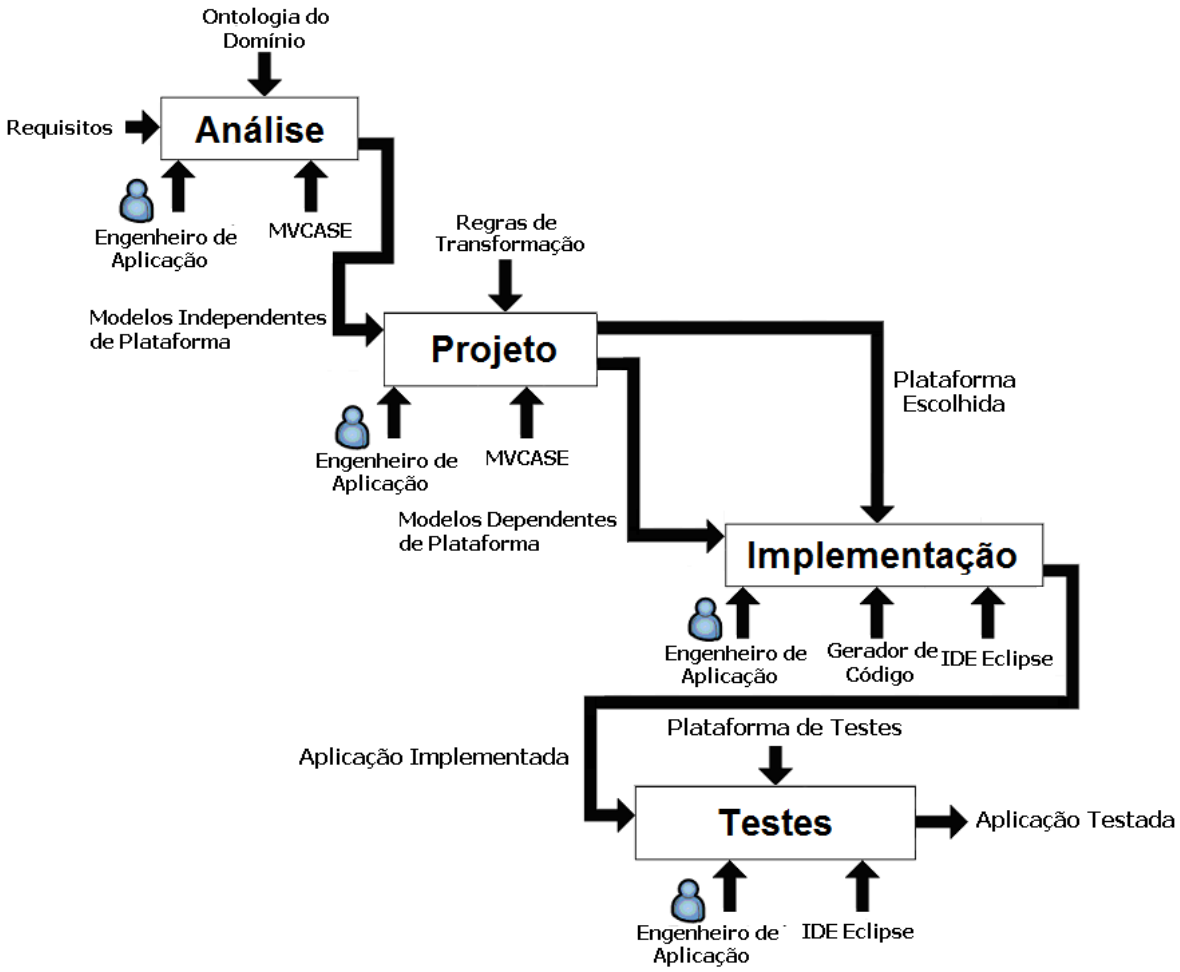


Figura 4.19. Engenharia de Aplicação.

As atividades dessa fase são facilitadas pelo reúso dos artefatos e ferramentas, construídos na Engenharia de Domínio. Nas disciplinas de Análise e no Projeto são utilizados os perfis na ferramenta MVCASE para a construção dos modelos independentes e dependentes de plataforma. Nessas disciplinas também são utilizadas as regras de transformação para a transformação dos modelos independentes de plataforma para os dependentes de plataforma.

Na disciplina de implementação são utilizados o gerador de código e os frameworks escolhidos para reúso na engenharia de domínio para facilitar a implementação das diferentes aplicações do domínio. Na disciplina de Testes, são utilizados os geradores de testes, que geram testes automatizados a partir de um modelo de projeto para uma plataforma de teste específica.

Nas próximas seções são apresentados os fluxos de trabalho das disciplinas da Engenharia de Aplicação, o estereótipo <<**ubicomp**>> é utilizado para destacar as atividades que atendem requisitos específicos da computação ubíqua, e o estereótipo <<**reuse**>> é utilizado para destacar as atividades onde são reutilizados ferramentas ou artefatos construídos na Engenharia de Domínio.

4.2.1. Análise

A Figura 4.20 ilustra o fluxo de trabalho da disciplina de análise. O primeiro passo, na atividade Analisar Requisitos, é definir os requisitos específicos da aplicação, isso é feito com base na análise da ontologia do domínio. No caso de uma aplicação ubíqua devem ser também definidos os requisitos relacionados com as suas adaptações de conteúdos e comportamentos, denominados Requisitos Adaptáveis.

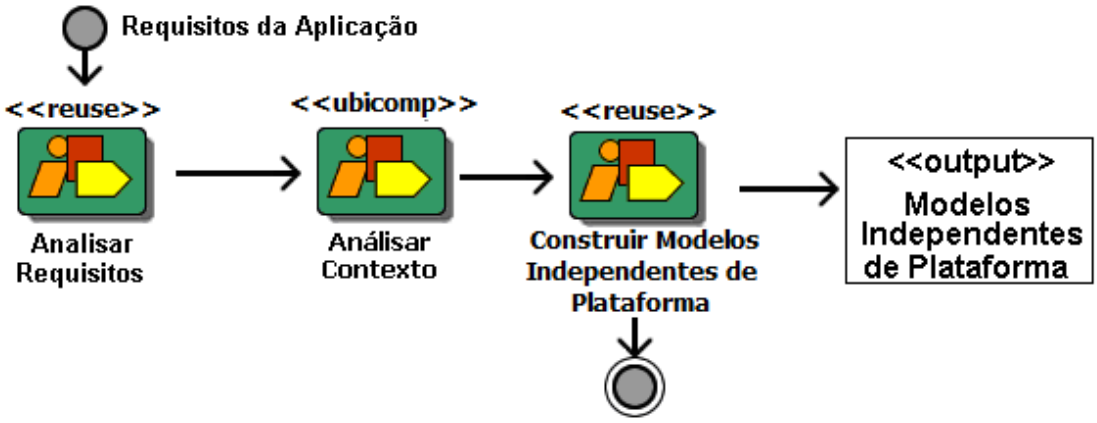


Figura 4.20. Fluxo de Trabalho na Disciplina de Análise.

Prosseguindo, especifica-se uma arquitetura, ainda abstrata, para a aplicação. Essa arquitetura orienta as próximas atividades da disciplina. Finalmente, com os requisitos específicos da aplicação refinados e analisados, são construidos os modelos independentes de plataforma (PIMs) baseados no perfil do domínio construídos na Engenharia de Domínio e integrados na ferramenta MVCASE.

4.2.2. Projeto

A Figura 4.21 apresenta o fluxo de trabalho da disciplina de projeto. Em Escolher Plataforma, são tomadas decisões sobre o sistema operacional, sistema gerenciador de banco de dados, linguagem de programação, modelos de dispositivos e outros padrões de desenvolvimento escolhidos para a implementação da aplicação. Essa escolha deve ser feita de acordo com as escolhas feitas na Engenharia de Domínio.

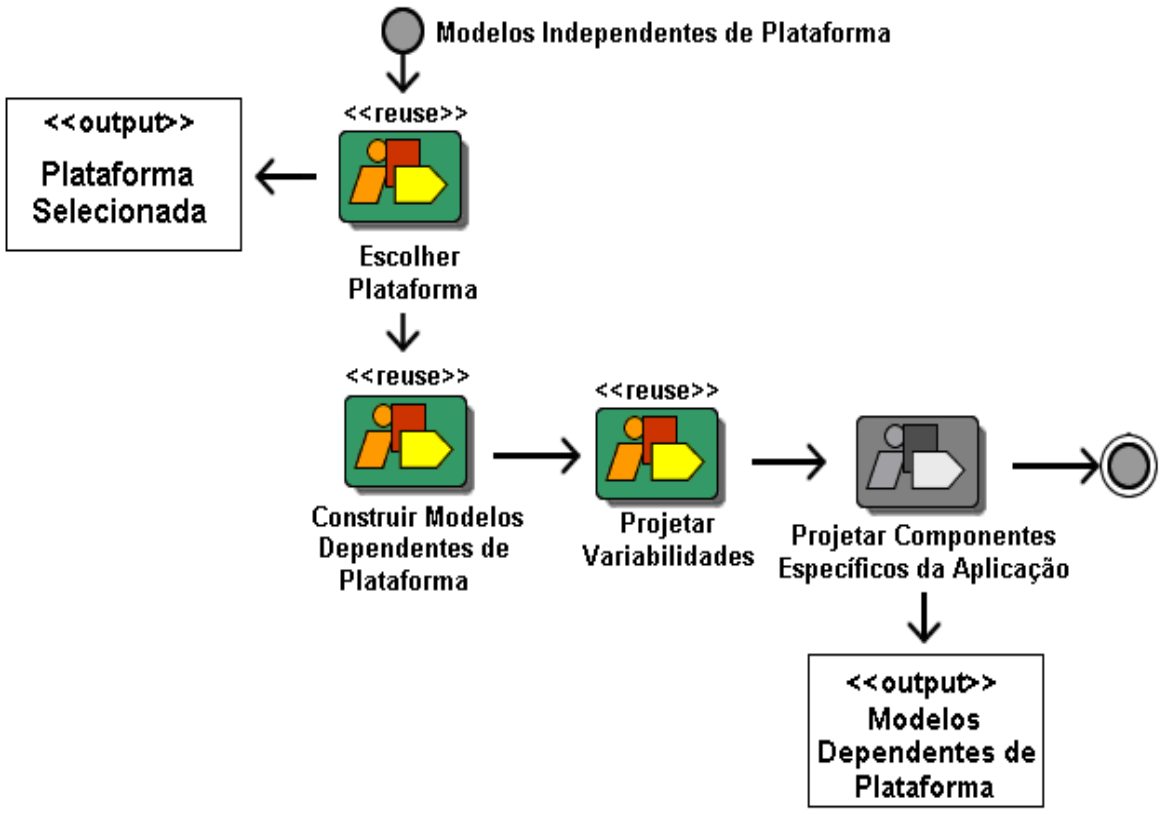


Figura 4.21. Fluxo de trabalho na disciplina de projeto.

Em Construir os Modelos Dependentes de Plataforma, parte-se dos modelos obtidos na Análise que, através de transformações automáticas baseadas nas regras de transformações definidas na Engenharia de Domínio, são transformados em modelos dependentes de plataforma (PSM), que contêm detalhes da plataforma escolhida, conforme as decisões de projeto. Esses modelos de classes de projeto servem de base para a geração de codigo na disciplina de implementação.

Na atividade Projetar Variabilidades as variabilidades são modeladas nos diagramas da aplicação do domínio, essas variabilidades são modeladas estendendo, principalmente os diagramas de classe e de sequência, para isso é utilizado o estereótipo <<variability>>. Na atividade Projetar Componentes Específicos da Aplicação os modelos da aplicação são estendidos para modelar também os componentes específicos da aplicação, para isso é utilizado o estereótipo <<application>>.

4.2.3. Implementação

A Figura 4.22 apresenta o fluxo de trabalho da disciplina de implementação. Na atividade Gerar Código, utilizando o gerador de código desenvolvido na engenharia de domínio é gerado o código fonte da aplicação, a partir de um modelo de aplicação baseado nos perfis definidos na engenharia de domínio. Em seguida, na atividade Implementar Componentes Específicos da Aplicação, são construídos os componentes dos requisitos específicos da aplicação.

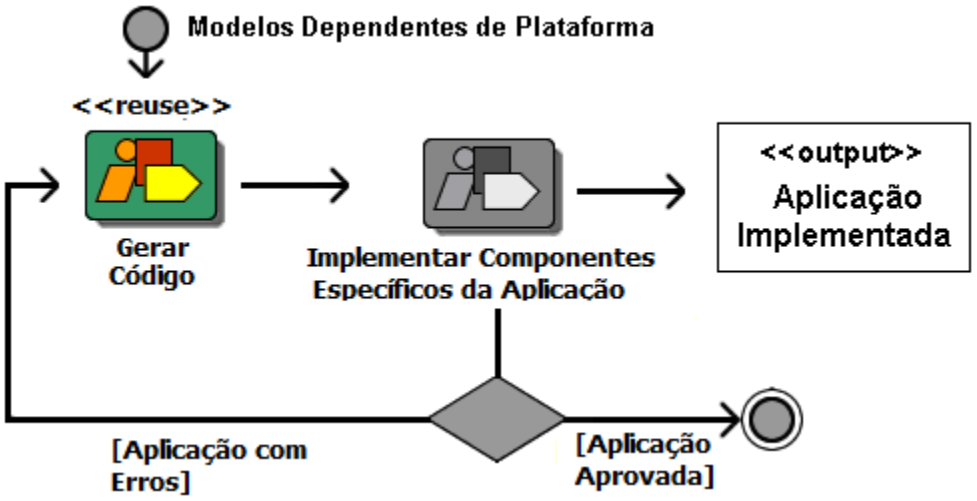


Figura 4.22. Fluxo de trabalho da disciplina de implementação.

4.2.4. Testes

A Figura 4.23 apresenta o fluxo de trabalho da disciplina de Teste. Na atividade Gerar Testes Automatizados, são gerados testes funcionais para uma plataforma de testes, como, por exemplo, o Selenium [KON07].

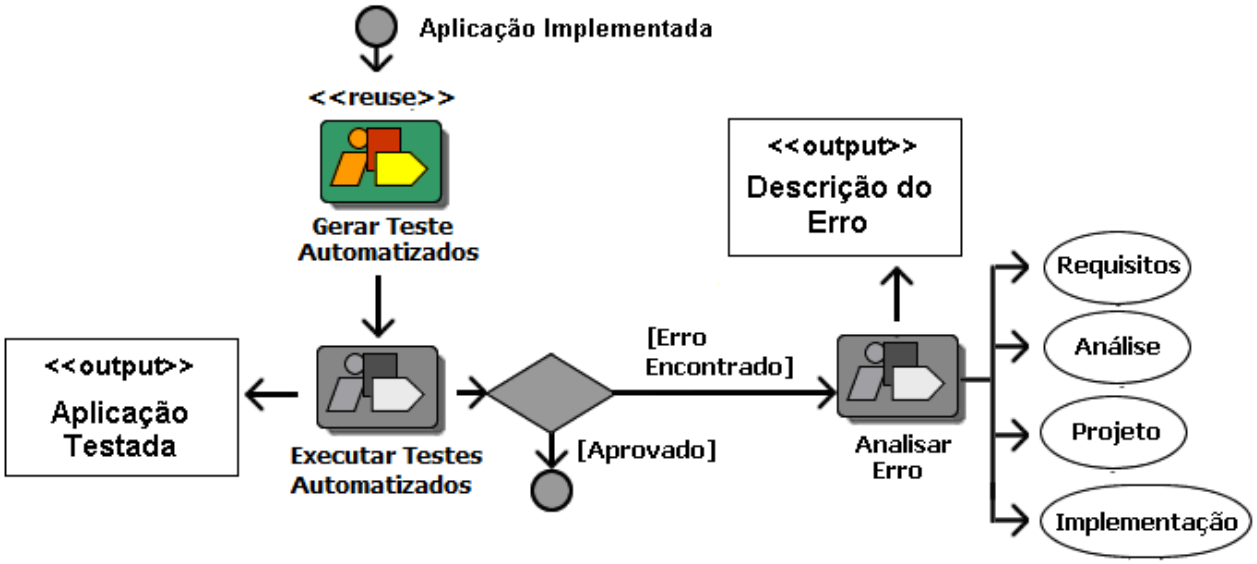


Figura 4.23. Fluxo de trabalho da disciplina de testes.

Em Analisar Erro, caso algum erro seja encontrado, esse erro é analisado, e a partir dessa análise, decide-se para qual fase do ciclo de desenvolvimento deve-se retornar para corrigir esse erro. Caso contrário, a aplicação é disponibilizada para a implantação.

4.2.5. 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 4.24 apresenta o fluxo de trabalho principal dessa disciplina, que tem início com o empacotamento da versão da aplicação a ser implantada. 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.

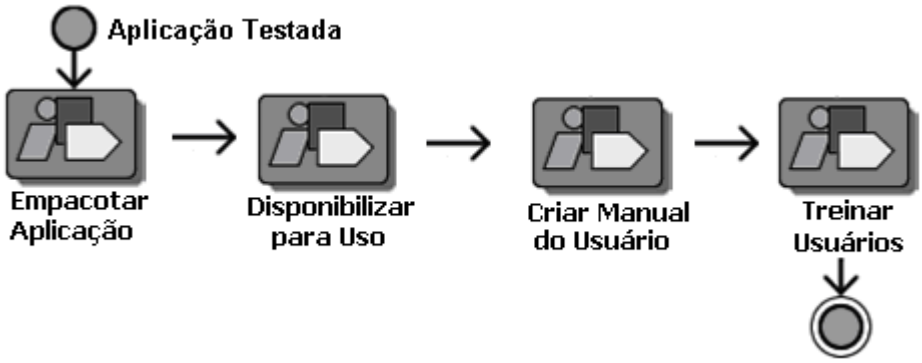


Figura 4.24. Fluxo de trabalho principal para disciplina Implantação.

4.3 Conclusões

Esse capítulo apresentou uma abordagem que sistematiza a utilização dos resultados da pesquisa apresentada nos capítulos anteriores. Essa abordagem tem foco no reúso de software, através principalmente da geração de código e da implementação de um framework específico de domínio para a implementação de aplicações pertencentes a um domínio específico.

Essa abordagem é composta por duas fases. Engenharia de Domínio, que evolui através das disciplinas de Análise do Domínio, Projeto do Domínio e Implementação do Domínio; e a Engenharia de Aplicação que evolui através das disciplinas de Requisitos, Análise, Projeto, Implementação, Testes e Implantação.

Cada uma das disciplinas das duas fases 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.

5. Estudo de Caso

Para exemplificar o uso da abordagem proposta, neste capítulo é apresentado o desenvolvimento de uma família de aplicações no domínio da educação médica construtivista que foi validado no curso de medicina da Universidade Federal de São Carlos.

Na Engenharia de Domínio esse domínio foi estudado e foram desenvolvidos os artefatos e as ferramentas que foram reutilizados no desenvolvimento das aplicações. Com apoio dos artefatos e ferramentas construídas na Engenharia de Domínio, foram identificadas quatro aplicações para esse domínio, das quais três foram implementadas, todas baseadas em uma aplicação principal, denominada Portfólio Reflexivo Eletrônico (PRE) [SAN09C] . As aplicações desenvolvidas foram: Uma versão do PRE, o MobilePRE, que é uma versão simplificada do PRE utilizada em dispositivos móveis e uma outra aplicação denominada P2PMobileLearning [SAN09D].

O PRE apóia o processo de ensino e aprendizagem no curso de Medicina da Universidade Federal de São Carlos (UFSCar). Esse curso foi criado em 2005 com uma proposta inovadora na organização curricular e nas atividades educacionais. Os estudantes, ao longo do curso, atuam em cenários diversificados, tais como domicílios, creches, escolas, Unidades de Saúde da Família e serviços hospitalares, no sentido de desenvolverem capacidades para o cuidado integral à saúde das pessoas e da comunidade. Na proposta pedagógica desse curso o processo de ensino-aprendizagem utiliza princípios da Aprendizagem Baseada em Problemas (ABP) [BLI95].

O MobilePRE é uma aplicação com um conjunto limitado das funcionalidades do PRE, e que é executado em dispositivos móveis. A P2PmobileLearning é uma aplicação integrada ao PRE que permite alunos e professores compartilharem conteúdos em uma rede Peer-to-Peer (P2P) em dispositivos móveis de diferentes capacidades (e.g. tamanho de tela, capacidade de processamento e armazenamento), através da adaptação de conteúdo e da ciência de contexto.

5.1. Engenharia de Domínio

Esta seção apresenta o desenvolvimento dos artefatos e ferramentas específicas do domínio do curso de medicina da UFSCar, que posteriormente na Engenharia de Aplicação, foram reutilizados no desenvolvimento das diferentes aplicações do domínio.

5.1.1. Análise do Domínio

Seguindo o processo proposto para a engenharia de domínio, a primeira disciplina é a análise do domínio Nessa atividade são identificados e modelados os requisitos do domínio, para isso foram realizadas diversas reuniões entre os engenheiros do domínio e alunos e professores do curso de medicina da UFSCar. Também foram acompanhadas atividades em sala de aula e atividades práticas dos alunos. A Figura 5.1 ilustra parte do documento de requisitos do domínio.

[RF001] Inserir Documentos

Descrição do caso de uso: Os alunos e professores devem ser capazes de inserir documentos no sistema, os documentos podem ser dos seguintes tipos: Nova Síntese, Sinteste Provisória e Avaliação. Esses documentos são descritos pelos seguintes campos: título, texto, tipo e autor.

Prioridade: ■ Essencial ☐ Importante ☐ Desejável

[RF002] Criar Tópico no Forúm

Descrição do caso de uso: Alunos e Professores devem ser capazes de criar um novo tópico no fórum, esse novo tópico deve ser listado em uma lista de tópicos e outros usuários devem ser capazes de responder esse tópico. O novo tópico é descrito pelos seguintes campos: título, texto e autor.

Prioridade: ☐ Essencial ■ Importante ☐ Desejável

Figura 5.1. Parte do Documento de Requisitos do Domínio.

A segunda atividade dessa disciplina é a modelagem dos requisitos em diagramas de caso de uso e em diagramas de tipos. A Figura 5.2 ilustra um diagrama de casos de uso de um requisito do domínio e a Figura 5.3 ilustra um diagrama de tipos do domínio.

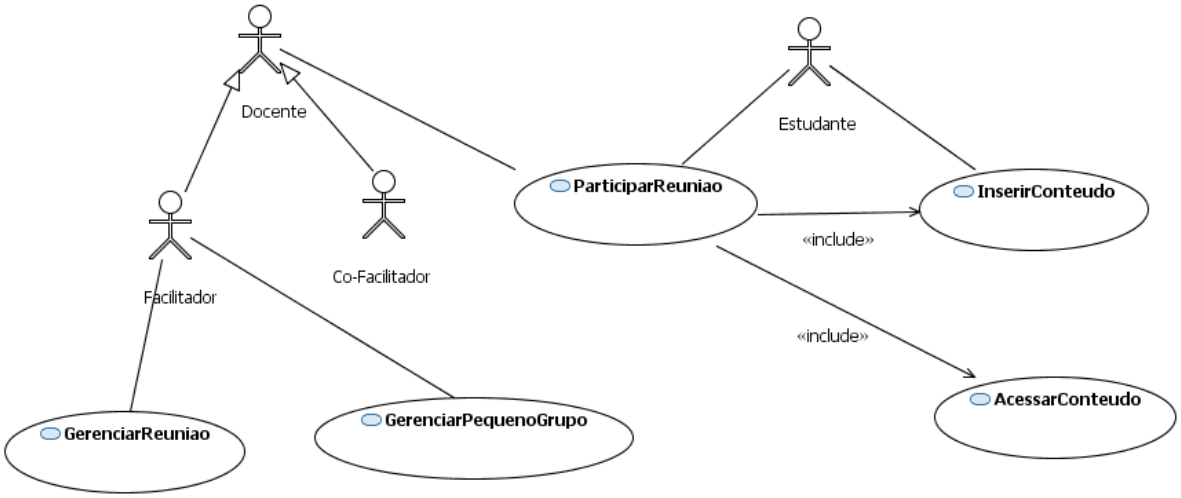


Figura 5.2. Casos de Uso do Domínio da Medicina da UFSCar.

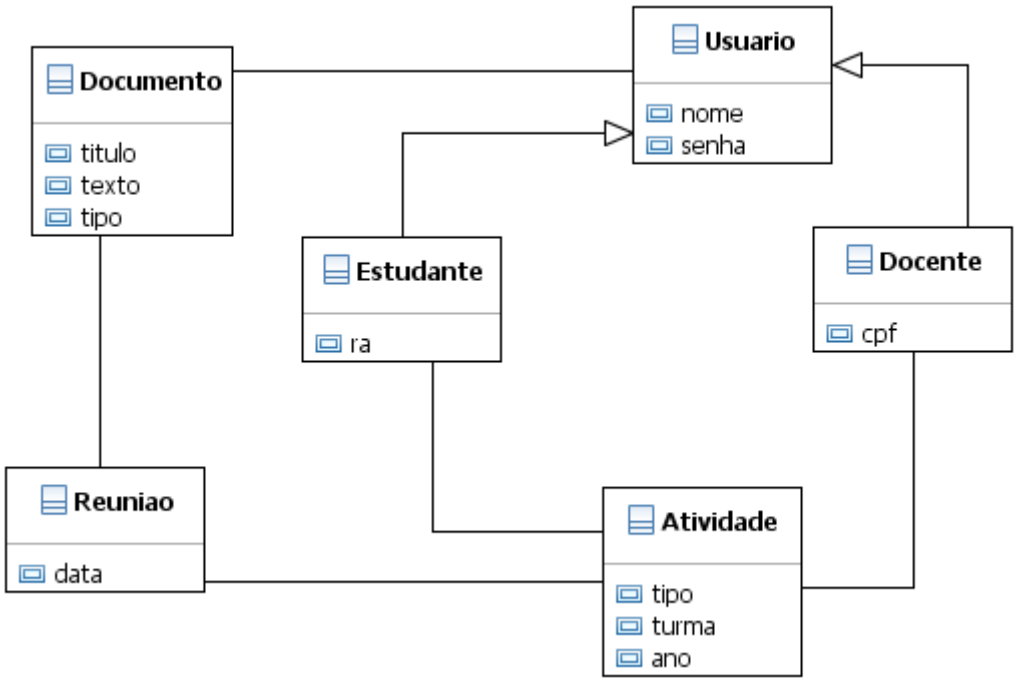


Figura 5.3. Diagrama de Tipos do Domínio.

Com os requisitos levantados e modelados, foi construída a ontologia que descreve o domínio. Essa ontologia serve como base para a construção do perfil que descreve o domínio. A Figura 5.4 apresenta parte dessa ontologia.

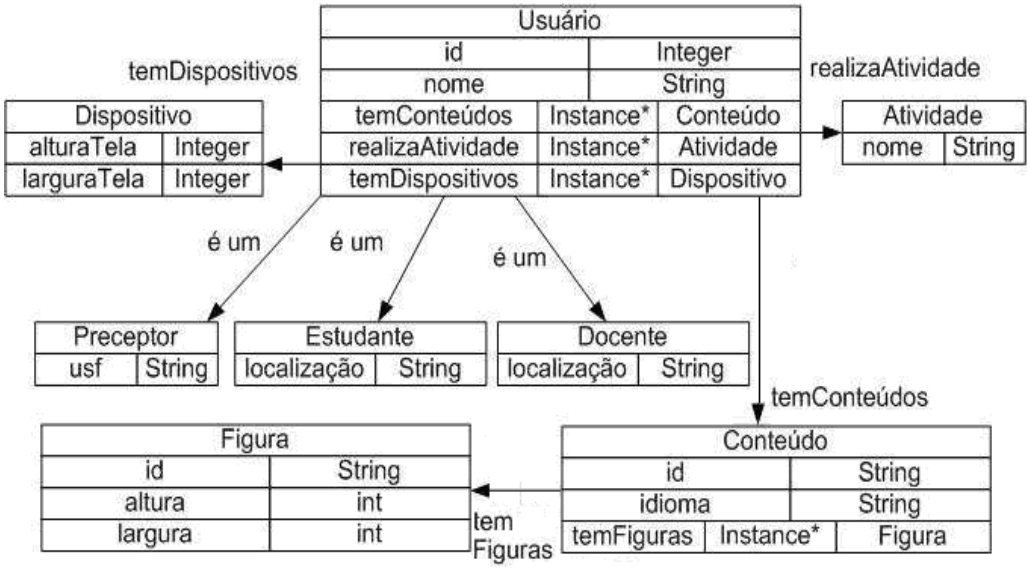


Figura 5.4. Parte da Ontologia do Domínio.

5.1.2. Projeto do Domínio

O primeiro passo do projeto do domínio é a definição das possíveis plataformas das aplicações do domínio. A Figura 5.5 apresenta o documento com a definição das plataformas do domínio.

Elemento	Plataform Utilizada	Versão
SGBD	PostgreSQL	8.3
Linguagem de Programação	Java	6.0
Plataformas	JEE e JME	5.0 e 2.0
SO	Linux e Windows	X

Figura 5.5. Documento de Definição de Plataforma.

Com o domínio descrito em uma ontologia, é possível refinar os modelos de análise para modelos de projeto do domínio. A Figura 5.6 apresenta o diagrama de classes de projeto do domínio.

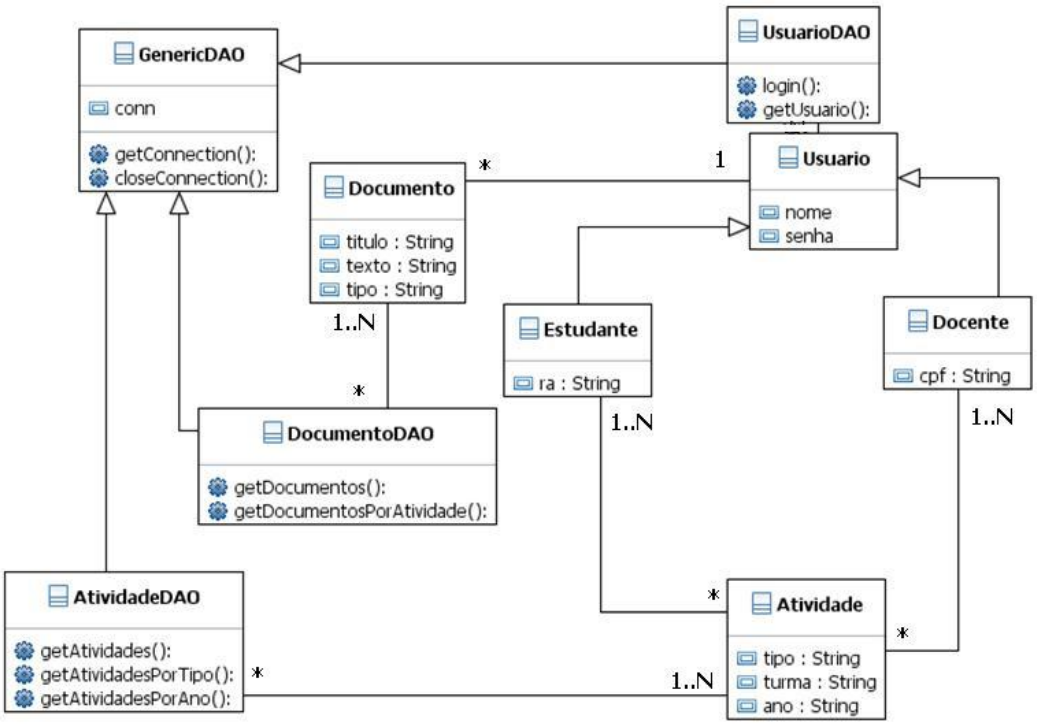


Figura 5.6. Diagrama de Classes de Projeto do Domínio.

A partir desses diagramas, utilizando o framework GMF, foi construído o perfil do domínio, que serve como base para a modelagem das aplicações do domínio. A Figura 5.7 apresenta parte da definição desse perfil.

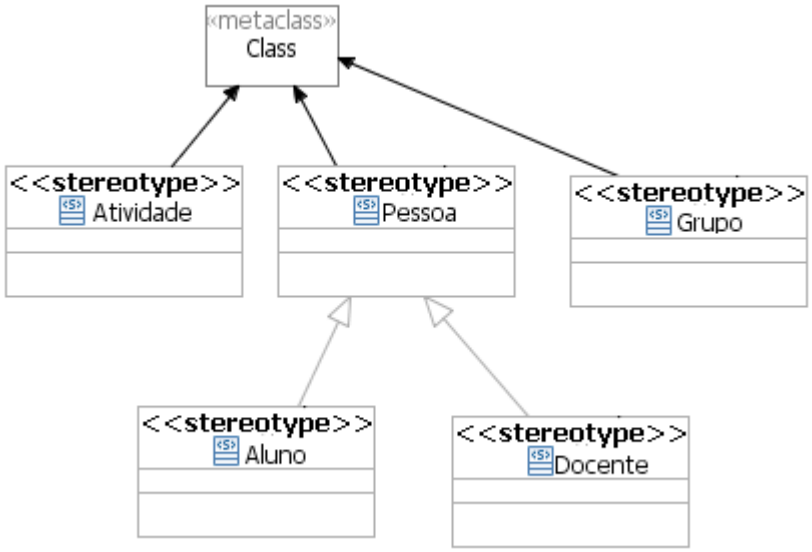


Figura 5.7. Parte do Perfil Domínio.

Depois de construído o perfil do domínio, são construídos os perfis para as plataformas de implementação definidas na atividade de definir plataformas. Nesse domínio foram definidos perfis para as plataformas JEE, JME e Android. A Figura 5.8, por exemplo, ilustra a definição do perfil da plataforma Android.

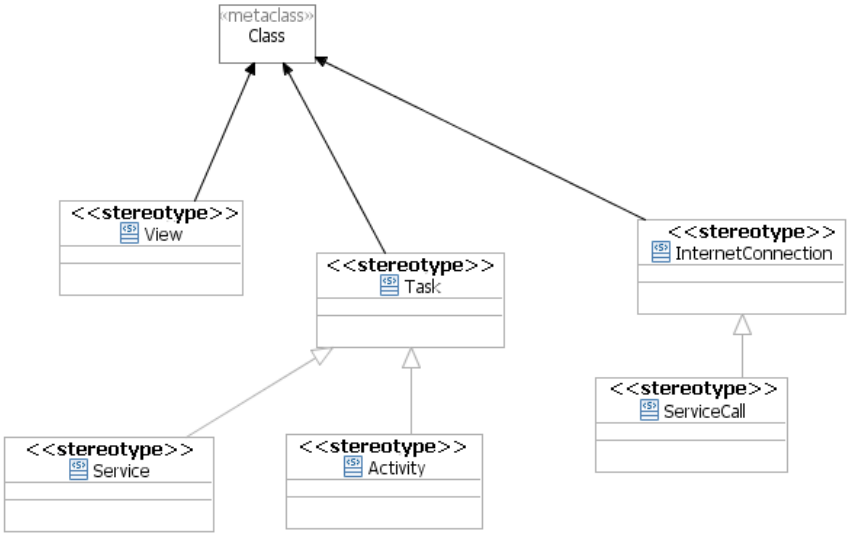


Figura 5.8. Parte do Perfil da Plataforma Android.

Depois são verificados os pontos variáveis do domínio e são tomadas as decisões de quais frameworks podem ser reutilizados no desenvolvimento das aplicações do domínio. A Figura 5.9 ilustra um documento que define os frameworks que serão utilizados no desenvolvimento das aplicações do domínio.

Framework	Versão	Funcão
UBICK	1.0	Ciência de Contexto e Adaptação de Conteúdo em Dispositivos Móveis
Hibernate	3.3.2	Persistência de Dados
Java Server Faces	1.2	Implementação do Padrão MVC
Jasper Reports	3.7	Geração de Relatórios
Rich Faces	3.3	Componentes de Visão

Figura 5.9. Frameworks Utilizados no Desenvolvimento das Aplicações.

5.1.3. Implementação do Domínio

A primeira atividade dessa disciplina é integrar os perfis do domínio e das plataformas na ferramenta MVCASE para facilitar a modelagem das aplicações do domínio. A Figura 5.10 apresenta uma tela da ferramenta MVCASE com um diagrama de classes utilizando o perfil do domínio.

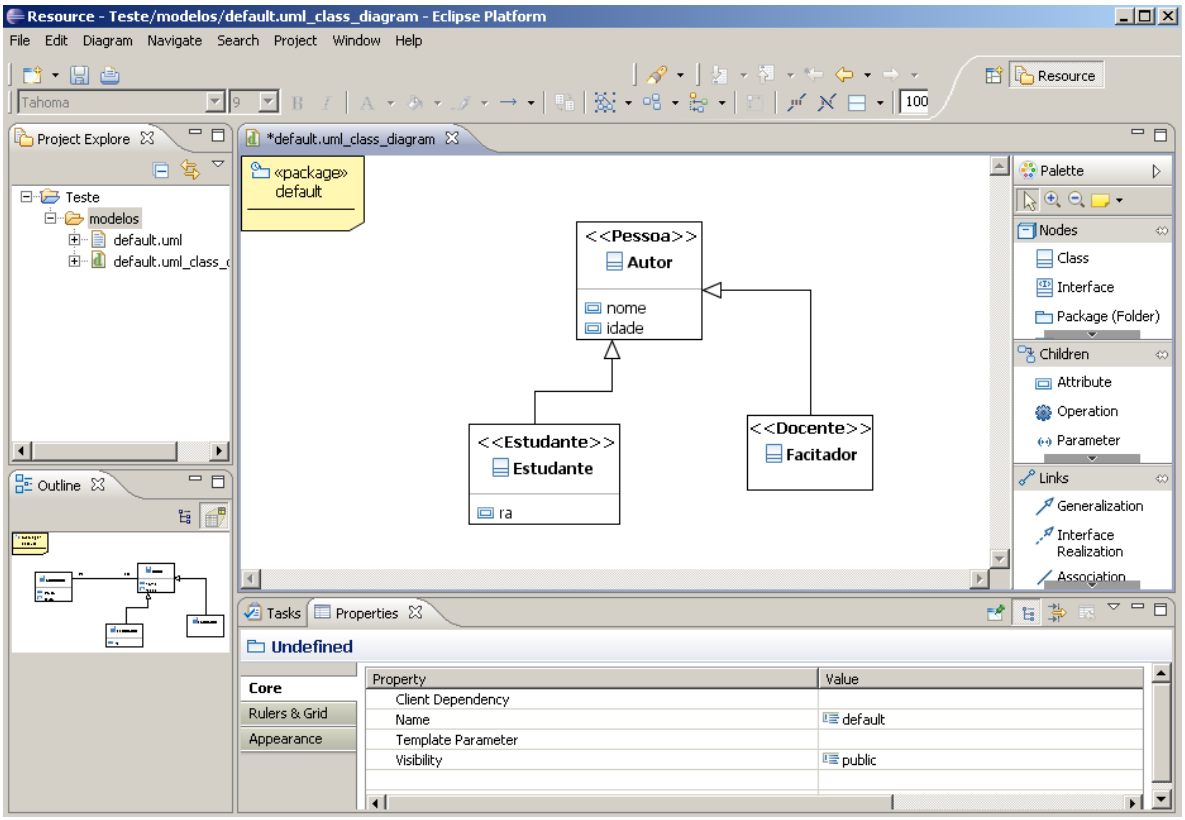


Figura 5.10. MVCASE com Perfil do Domínio.

Na Implementação do Domínio, também foi implementado o gerador de código, para as plataformas JEE, JME e Android. A Figura 5.11 apresenta parte da descrição dos templates no framework JET.

```
<%@taglib prefix="ws" id="org.eclipse.jet.workspaceTags" %>

<ws:file template="templates/dao/GenericDAO.jet" path="{$org.eclipse.jet.resource
<ws:file template="templates/controller/GenericController.jet" path="{$org.eclipse


<ws:file template="templates/beans/Pessoa.jet" path="{$org.eclipse.jet.resource.p
<ws:file template="templates/dao/PessoaDAO.jet" path="{$org.eclipse.jet.resource.
<ws:file template="templates/controller/PessoaController.jet" path="{$org.eclipse

<ws:file template="templates/beans/Autor.jet" path="{$org.eclipse.jet.resource.pr
<ws:file template="templates/dao/AutorDAO.jet" path="{$org.eclipse.jet.resource.p
<ws:file template="templates/controller/AutorController.jet" path="{$org.eclipse.

<c:iterate select="/Model/ownedElement" var="currClass" >
    <c:if test="$currClass[self::Estudante]">
        <ws:file template="templates/beans/Estudante.jet"   path="{$org.eclipse.je
        <ws:file template="templates/dao/EstudanteDAO.jet"  path="{$org.eclipse.je
    </c:if>
</c:iterate>

<c:iterate select="/Model/ownedElement" var="currClass">
    <c:if test="$currClass[self::Facilitador]">
```

Figura 5.11. Template para a Geração de Código na Plataforma JEE.

Além da ferramenta CASE e do Gerador de Código, na implementação do domínio são descritas as regras de transformações entre os modelos independentes de plataforma para os modelos dependentes de plataforma, para isso foi utilizado o framework Eclipse ATL. A Figura 5.12 apresenta algumas das regras de transformação definidas.

```
rule Model2Model {

    from s:pre!Model
    to t:psm!JModel(
        ownedElement <- s.ownedElement
    )

}

rule Facilitador2JFacilitador {

    from s:pre!Facilitador
    to t:psm!JFacilitador(
        name <- s.name,
        type <- 'Entity,View',
        ownedElement <- s.ownedElement
    )
}
```

```
rule Estudante2JEstudante {

    from s:pre!Estudante
    to t:psm!JEstudante(
        name <- s.name,
        type <- 'Entity,View',
        ownedElement <- s.ownedElement
    )
}
rule Conteudo2JConteudo {

    from s:pre!Conteudo
    to t:psm!JConteudo(
        name <- s.name,
        type <- 'Entity, View, ServiceProvider',
        ownedElement <- s.ownedElement
    )
}
```

Figura 5.12. Regras de transformação do PIM para o PSM.

Depois de desenvolvidas todas as ferramentas do domínio é possível implementar mais facilmente e rapidamente as diferentes aplicações pertencentes ao domínio. As próximas seções apresentam o processo de desenvolvimento detalhado do PRE, e no fim apresenta o resultado final das outras duas aplicações desenvolvidas (P2PmobileLearning e MobilePRE).

5.2. Engenharia de Aplicação

Na engenharia de aplicação são desenvolvidas as diversas aplicações do domínio com a reutilização dos artefatos e ferramentas desenvolvidos na Engenharia de Domínio (Ontologia, Perfis UML, Regras de Transformação e Gerador de Código e de Testes). Essa seção apresenta o desenvolvimento completo de uma versão do PRE e também mostra o resultado final de outras duas aplicações do domínio implementadas com a abordagem proposta.

5.2.1. Análise

No desenvolvimento do PRE, partindo das ontologias construídas na engenharia de domínio, foram levantados os requisitos específicos da aplicação. A Figura 5.13 apresenta parte do documento de requisitos do PRE.

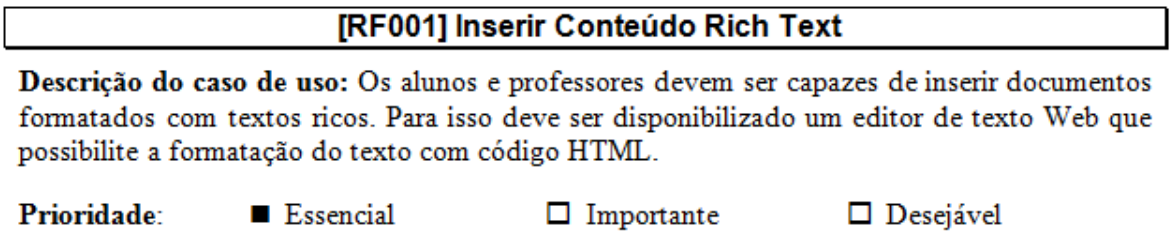


Figura 5.13. Requisito Específico da Aplicação.

Depois de elicitados os requisitos são modelados em diagramas de casos de uso. A Figura 5.14 apresenta um caso de uso de um requisito específico da aplicação.

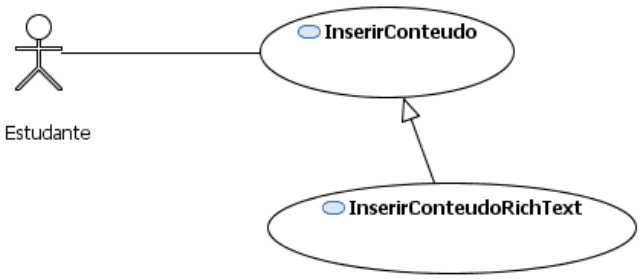


Figura 5.14. Caso de Uso Específico da Aplicação.

Posteriormente, os casos de uso são detalhados em modelos de sequência, e os objetos são modelados em diagramas de Tipos. A Figura 5.15 mostra, por exemplo, um modelo de classes do PRE, utilizando o perfil do domínio definido na Engenharia de Domínio.

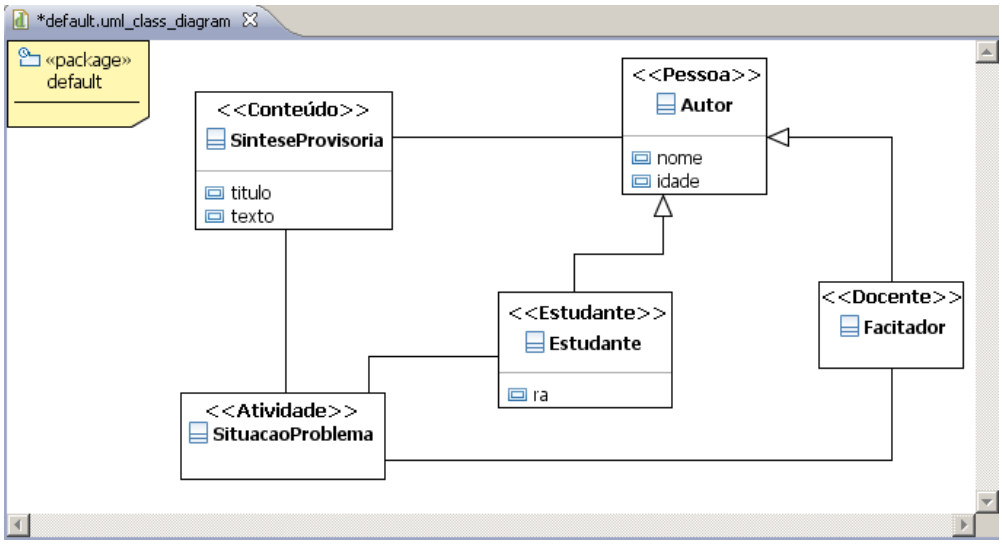


Figura 5.15. Modelagem do PRE na Análise.

5.2.2. Projeto

Nessa fase é definida a plataforma de implementação da aplicação. No caso do PRE, foi adotada a plataforma JEE, a linguagem Java e o SGBD PostgreSQL. A Figura 5.16 mostra um modelo de classes de projeto obtido através da transformação do modelo de análise.

Nesse modelo os estereótipos Model, View e Controller representam o perfil da plataforma JEE desenvolvido na engenharia de domínio.

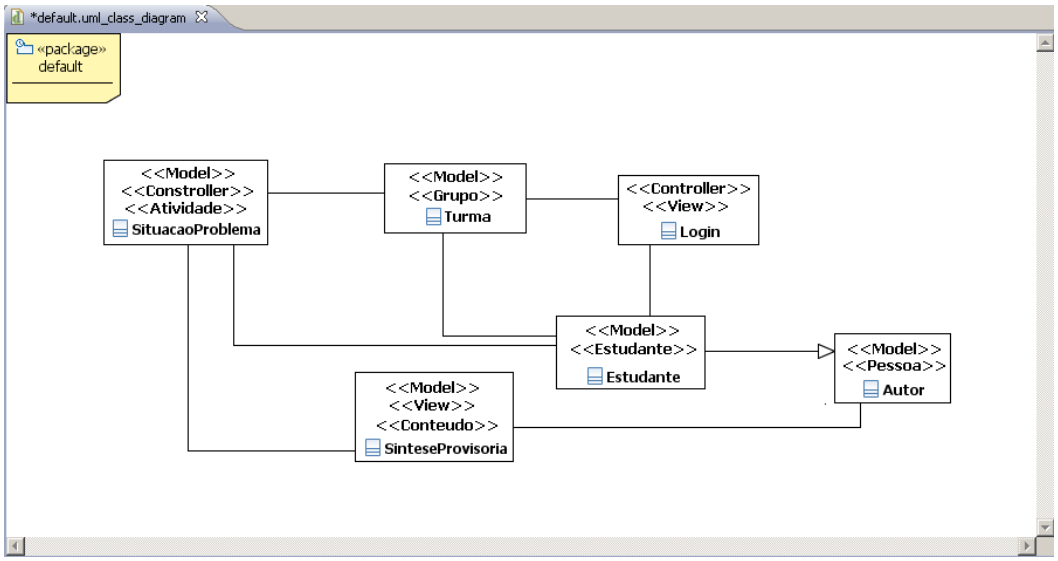


Figura 5.16. Modelagem do PRE na fase de Projeto.

Depois do projeto da aplicação são modelados os componentes específicos da aplicação e as variabilidades. A Figura 5.17 ilustra o modelo de classes da aplicação com os componentes específicos e as variabilidades modeladas.

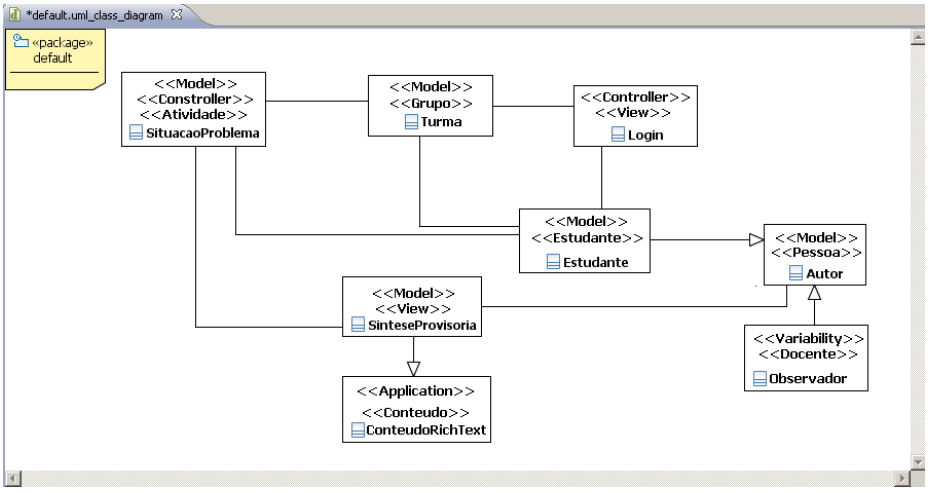


Figura 5.17. Diagrama de Classes com Variabilidades e Componentes Específicos da Aplicação Modelados.

5.2.3. Implementação

A partir dos modelos de projeto foi gerada grande parte do código da aplicação. Algumas funcionalidades como criação e acesso a conteúdos e o controle de acesso são totalmente gerados, outras são parcialmente geradas, necessitando de alterações manuais no código. Como a ferramenta desenvolvida é um plugin da IDE Eclipse, o desenvolvedor pode gerar e refinar o código da aplicação diretamente nesta IDE até sua versão final. A Figura 5.18 ilustra uma tela da IDE Eclipse com um código gerado pela ferramenta.

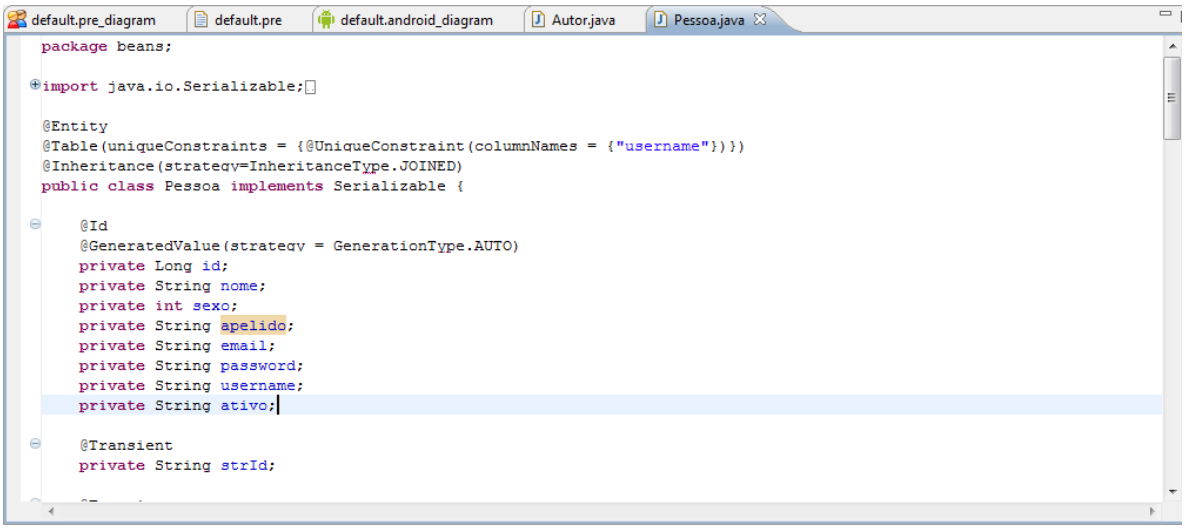


Figura 5.18. Código gerado para a plataforma JEE.

Depois da geração do código, são desenvolvidos os componentes específicos da aplicação e esses componentes são integrados nos pontos variáveis do código gerado.

5.2.4. Testes

Nessa disciplina foi utilizado o framework Selenium, que realiza testes automatizados em aplicações Web. A Figura 5.19 apresenta uma tela com a definição de um teste gerado para o framework selenium.

```
import com.thoughtworks.selenium.*;
import java.util.regex.Pattern;

public class LoginTeste extends SeleneseTestCase {
    public void login() throws Exception {
        selenium.open("/");
        selenium.type("Login.username", "admin");
        selenium.type("Login.password", "administrator");
        selenium.selectFrame("index");
        selenium.click("form:tree:applications:enterpriseApplications:enterpriseApplications_link");
        selenium.selectFrame("relative=up");
        selenium.selectFrame("main");
        verifyTrue(selenium.isTextPresent("No items found."));
        selenium.selectFrame("relative=up");
        selenium.selectFrame("header");
        selenium.click("propertyForm:Masthead:logoutLink");
        assertEquals("Log Out of Application Server Admin Console", selenium.getConfirmation());
        selenium.click("loginButton");
    }
}
```

Figura 5.19. Definição de um Caso de Teste no Framework Selenium.

5.2.5. Implantação

O PRE ainda não foi implantado para produção. A aplicação está temporariamente implantada em um servidor do Departamento de Computação, apenas para a realização dos testes com os grupos piloto do curso de Medicina. Nessa versão de testes o *deployment* do PRE foi realizado no servidor JBoss [JAM09] versão 4.2.2 e com o Servidor de Banco de Dados Postgres versão 8.3 [TEA00].

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 é necessário criar materiais de apoio, e, possivelmente, realizar sessões de treinamento para os envolvidos. As Figuras 5.20 e 5.21 apresentam imagens do PRE sendo utilizado por alunos do curso de medicina.

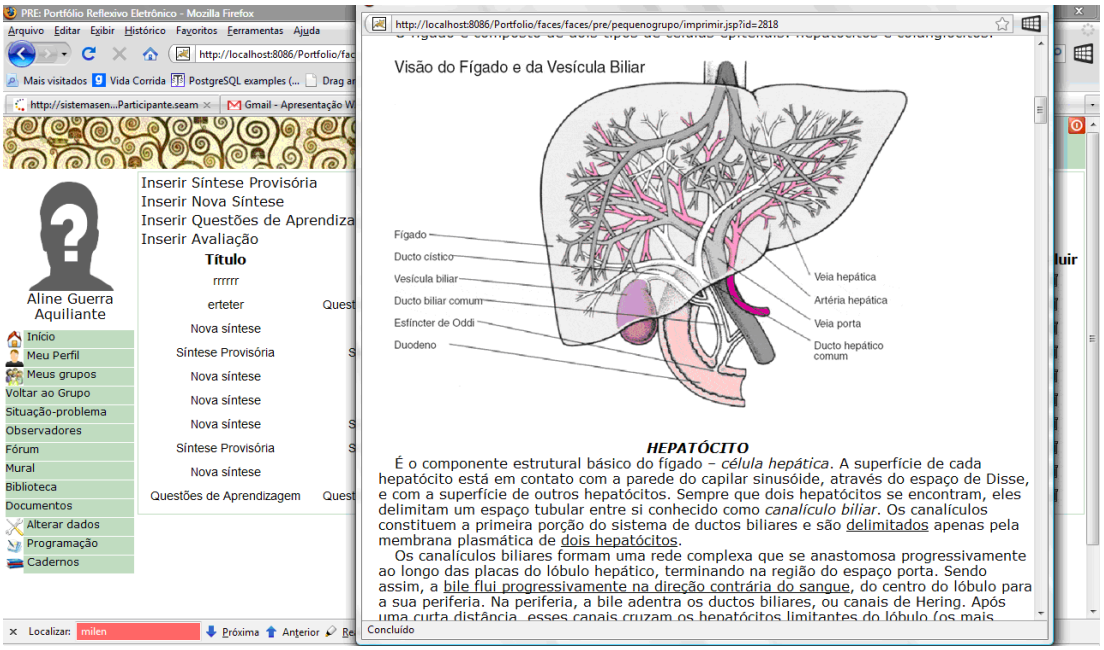


Figura 5.20. Tela do PRE.

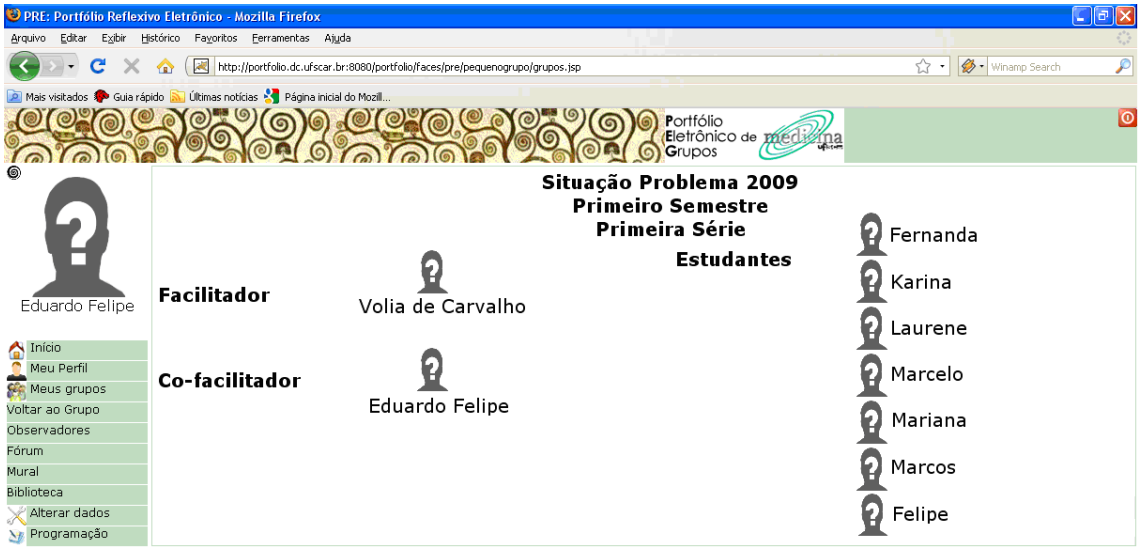


Figura 5.21. Tela do PRE.

5.3. Avaliação do PRE

Para a realização da avaliação, o protótipo do PRE foi disponibilizado no ambiente de uso, conforme ilustra a Figura 5.22. Foram constituídos dois grupos piloto na atividade Estação de Simulação [VAR08], ambos formados 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 2009, num ambiente que contou com um laptop para cada participante do grupo e câmeras e microfones para o registro dessa atividade.

Figura 5.22. Grupo Piloto do PRE.

Para avaliar o uso do ambiente, foi preparado um questionário com seis perguntas que visaram avaliar a satisfação dos alunos em relação à aplicação.

A Figura 5.23 ilustra os resultados da afirmação "O uso de tecnologia pode facilitar o processo de ensino aprendizagem em um curso de medicina". 57.1% dos estudantes responderam que estão totalmente de acordo, enquanto que 28,5% responderam que concordam parcialmente e 14,4% responderam que não tem opinião formada sobre o assunto.

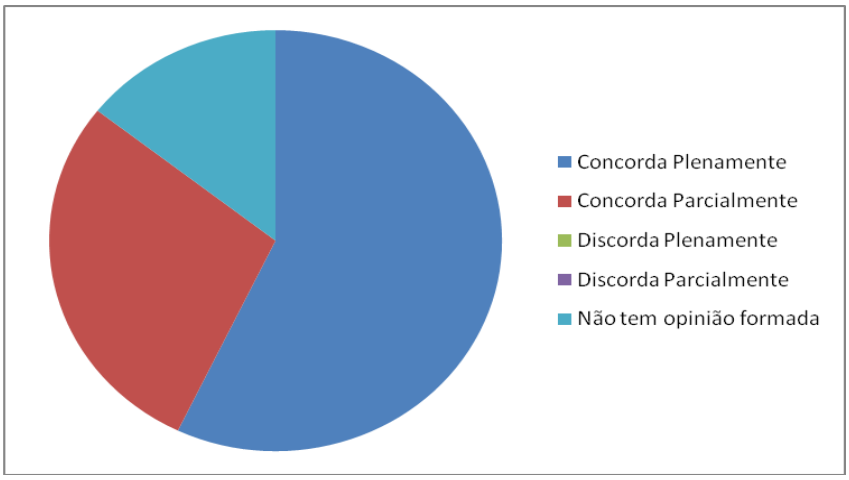


Figura 5.23. Resultados da questão "O uso de tecnologia pode facilitar o processo de ensino aprendizagem em um curso de medicina".

A Figura 5.24 ilustra os resultados da afirmação "O uso do PRE atrapalhou as atividades do meu grupo". 42.8% discordam plenamente da afirmativa, 28.6% responderam que discordam parcialmente da afirmativa, e 28.6% responderam que concordam parcialmente com a afirmativa.

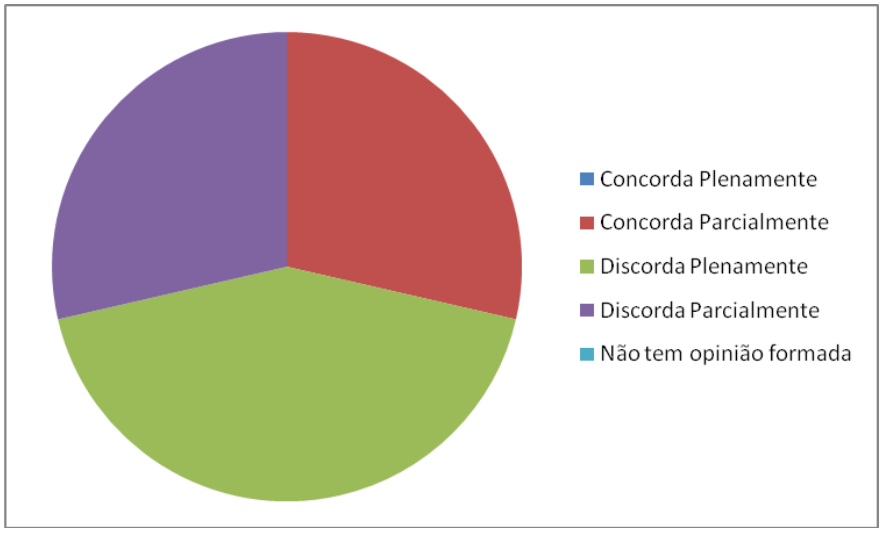


Figura 5.24. Resultado da questão "O uso do PRE atrapalhou as atividades do meu grupo"

A Figura 5.25 ilustra os resultados da questão "O PRE atrapalhou minhas atividades individuais". 14.3% responderam que discordam plenamente da afirmação, 57.2%

responderam que discordam parciamente da afirmação, e 28.5% responderam que concordam parcialmente com a afirmação.

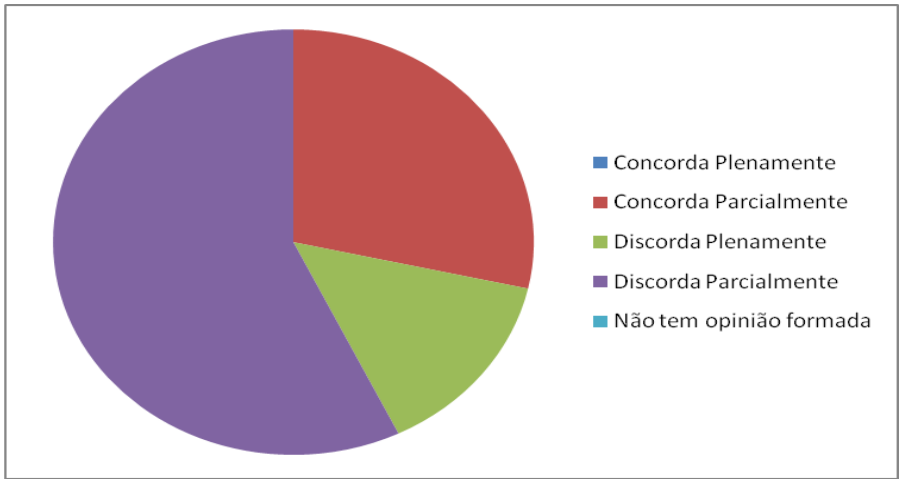


Figura 5.25. Resultado da questão "O PRE atrapalhou minhas atividades individuais".

A Figura 5.26 apresenta os resultados para a questão "Ainda que tenha problemas, o PRE pode vir a facilitar as atividades de ensino aprendizagem do curso de medicina da UFSCar". 57.1% responderam que concordam plenamente com a afirmação, 42.9% responderam que concordam parcialmente com a afirmação.

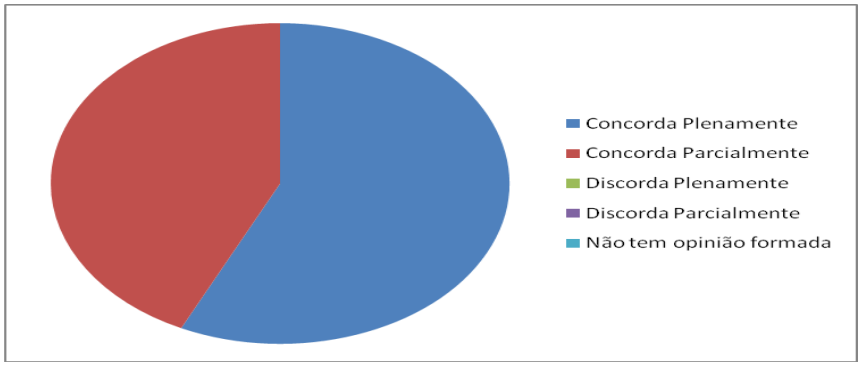


Figure 5.26. resultado da questão "Ainda que tenha problemas, o PRE pode vir a facilitar as atividades de ensino aprendizagem do curso de medicina da UFSCar".

A Figura 5.27 ilustra os resultados de ranqueamento que definem quais são as maiores vantagens do PRE na percepção dos participantes. Segundo essa pesquisa, a maior vantagem é a facilidade outorgada pelo PRE durante os estudos, a facilidade de comunicação entre os alunos , na terceira posição esta o enriquecimento das aulas com conteúdo multimedia e na quarta a facilidade de locomoção .

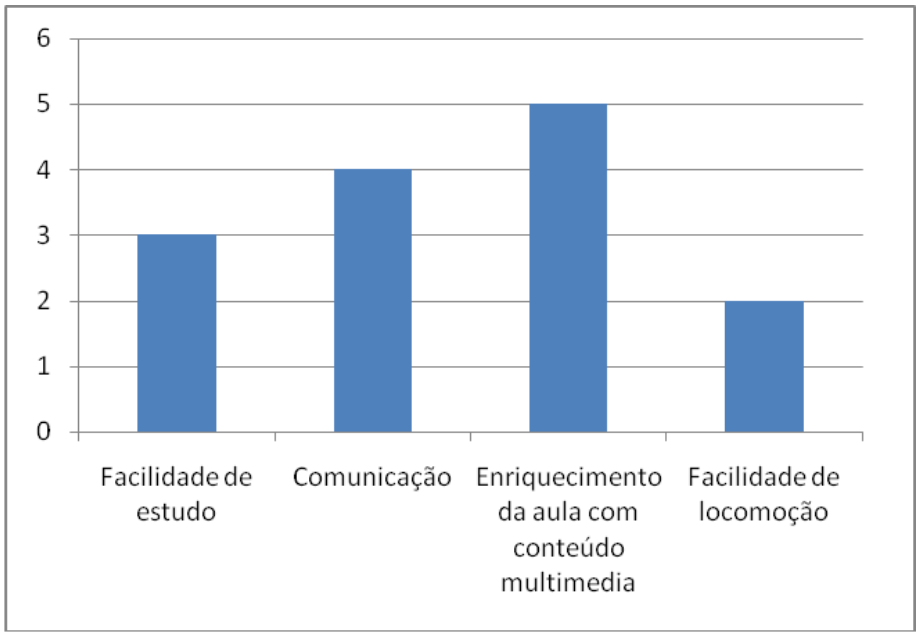


Figura 5.27. Ranqueamento das vantagens do PRE.

A Figura 5.28 apresenta os resultados para o ranqueamento que definem quais são as maiores desvantagens, na percepção dos participantes, do uso do PRE. Segundo essa pesquisa, o maior problema foram dificuldades no uso da aplicação,o segundo foi falta de treinamento, o terceiro foram erros de comunicação, e a quarta foram empatados erros da aplicação e dificuldade no uso de computadores.

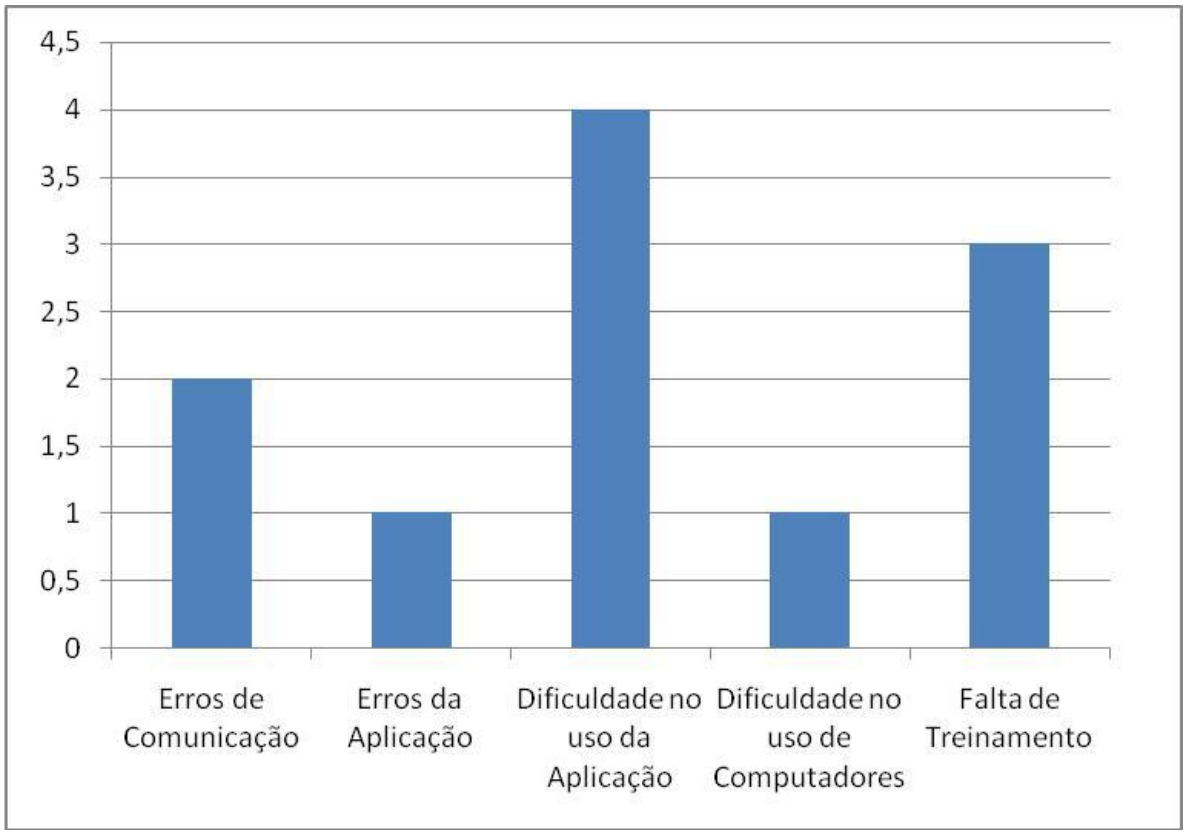


Figure 5.28. Ranqueamento das Desvantagens do PRE.

5.4. Outras Aplicações do Domínio

Além do PRE foram implementadas outras duas aplicações com o apoio das ferramentas desenvolvidas na Engenharia de Domínio, o MobilePRE, desenvolvido na plataforma Android , e que é uma versão com um conjunto limitado das funcionalidades do PRE, para ser acessada a partir de dispositivos móveis. A Figura 5.29 apresenta a modelagem dessa aplicação na disciplina de projeto na plataforma Android. A Figura 5.30 apresenta uma tela da aplicação no simulador da plataforma Android.

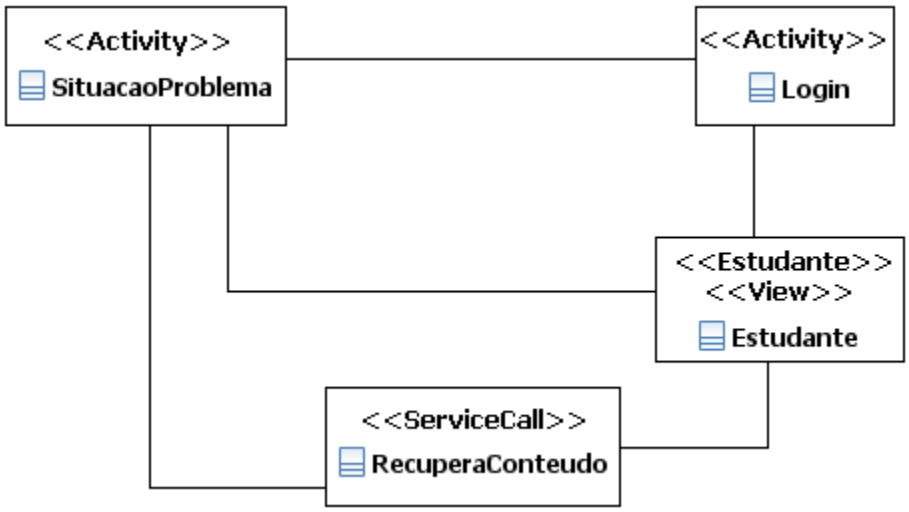


Figura 5.29. Parte da Modelagem de Projeto da Aplicação MobilePRE.

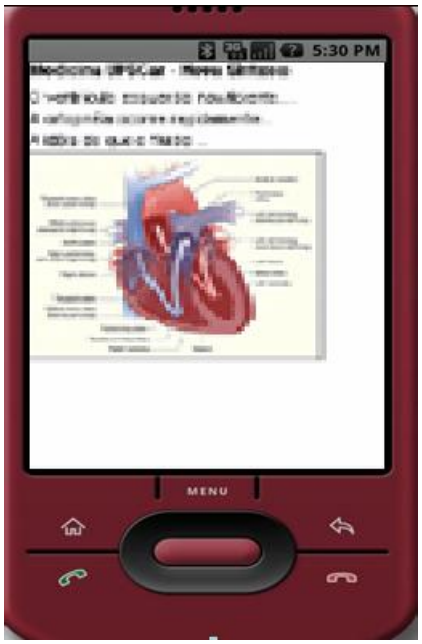


Figura 5.30. MobilePRE

A outra aplicação desenvolvida foi a P2PmobileLearning [SAN09 5], ilustrada na Figura 5.31. Essa aplicação é utilizada para alunos e professores do curso de medicina que realizam atividades fora da universidade e que devem compartilhar seus conteúdos. Essa aplicação compartilha conteúdo entre dispositivos de diferentes capacidades através da adaptação de conteúdo. Essa aplicação foi desenvolvida com a plataforma JME [JME10].

Figura 5.31. P2PMobileLearning.

6. Estudo de Viabilidade da Abordagem de Desenvolvimento de Software

O estudo de viabilidadedo processo de desenvolvimento foi feita de acordo com levantamentos sobre qualidade e esforço feitos durante o desenvolvimento e uso do PRE. Alguns dados como quantidade de artefatos e código reutilizados, tempo de desenvolvimento e facilidade na manutenção das aplicações foram levantados e comparados a dados levantados no desenvolvimento da primeira versão do PRE [SAN08].

6.1. Tempo de Desenvolvimento

As Figuras 6.1 e 6.2 ilustram o tempo gasto nos desenvolvimentos de duas versões (1 e 2) do PRE para dispositivos móveis. A primeira, desenvolvida por um Engenheiro de software e um programador, sem reúso do conhecimento do domínio ou gerador de código. A segunda, desenvolvida por apenas um Engenheiro de Software, nos papéis de Engenheiro de Domínio e de Aplicação, seguindo a abordagem proposta.

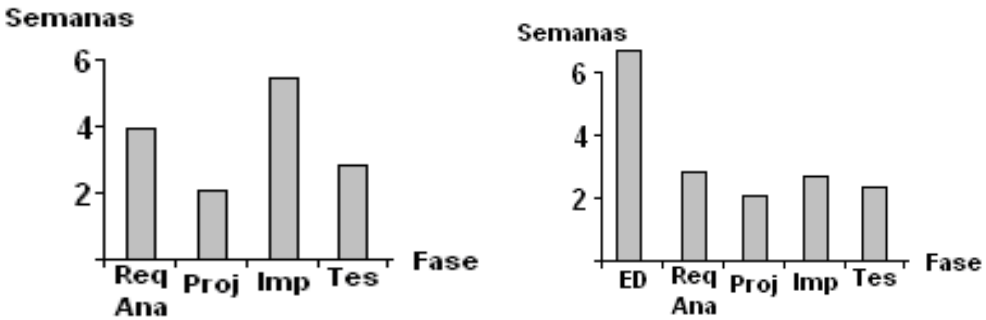


Figura 6.1. Tempo de Desenvolvimento das duas versões do PRE.

Conforme mostram os gráficos verifica-se que o tempo total do desenvolvimento da versão 1 foi de 13 semanas, e o da versão 2 de 15 semanas. Portanto, o processo da abordagem proposta (versão 2) gastou 2 semanas a mais, dado principalmente ao esforço na

fase da Engenharia de Domínio. Contudo, verifica-se que no desenvolvimento, especificamente da aplicação PRE, na abordagem proposta, o tempo de desenvolvimento foi de apenas 8 semanas. Essa redução no tempo se justifica considerando que grande parte do desenvolvimento da aplicação baseou-se na reutilização dos metamodelos construídos na Engenharia de Domínio e que grande parte do código foi gerado.

Portanto, uma maior vantagem do uso da abordagem proposta vem quando se desenvolvem novas aplicações para o mesmo domínio. Uma vez desenvolvida a engenharia de domínio, outras aplicações podem ser desenvolvidas mais facilmente e rapidamente, do que usando o processo normal sem a DSM e o gerador de código. Outra vantagem do processo proposto vem com a diminuição dos erros de modelagem e de implementação considerando o reúso dos metamodelos e a geração correta de código.

6.2. Reúso

A primeira versão do PRE foi desenvolvida em um processo, que não contava com a fase de engenharia de domínio. O único artefato reutilizado foi o framework UBICK utilizado também no desenvolvimento da segunda versão do PRE. A Tabela 6.1 apresenta alguns dados sobre a reutilização de artefatos no desenvolvimento das duas versões do PRE.

Tabela 6.1. Reúso no desenvolvimento do PRE.

Artefato/Versão	V1	V2
Framework	Foi reutilizado o framework UBICK.	Foi reutilizada a extensão do Framework UBICK.
Geração de Código	Apenas Classes, Atributos, Relacionamentos e Assinatura de Métodos a partir de diagramas de classe da UML.	Funcionalidades completas ou parciais a partir dos modelos específicos do domínio baseados nos perfis do domínio e da plataforma.
Testes Automatizados	-	Testes gerados para o framework Selenium a partir dos modelos que descrevem as aplicações do domínio

A partir dos resultados ilustrados na tabela, verifica-se que no desenvolvimento da segunda versão da aplicação a quantidade de artefatos reutilizados foi maior do que no desenvolvimento da primeira versão, o que ajuda a explicar a redução no tempo de desenvolvimento da aplicação, apresentado na seção 6.1.

6.3. Manutenção

A manutenção da nova versão do PRE é mais fácil do que na versão anterior, por dois motivos. Em primeiro lugar, a maior parte da manutenção da aplicação, principalmente inclusão de novas funcionalidades, é feita principalmente em modelos, mas fáceis de alterar do que código fonte.

Em segundo lugar, o código gerado deve seguir padrões de desenvolvimento, que garantem uma maior qualidade no código fonte e facilitam futuras manutenções que são realizadas diretamente no código fonte da aplicação, visando principalmente a correção de bugs.

7. Conclusões

Esta dissertação apresentou uma abordagem orientada por modelos para o desenvolvimento de software na computação ubíqua baseada na Modelagem Específica de Domínio e na Arquitetura Orientada por Modelos. As experiências têm sido realizadas no domínio de aplicações ubíquas no ensino de medicina. A abordagem proposta é composta de duas fases, Engenharia de Domínio e Engenharia de Aplicação.

Na Engenharia de Domínio, baseado no reúso de software e utilizando o conceito de modelagem especifica de domínio é elaborado um perfil UML que descreve um domínio específico e é implementado um gerador de código orientado para uma plataforma de implementação específica. Na Engenharia de Aplicação, baseado no perfil do domínio, e com o apoio de ferramentas desenvolvidas na Engenharia de Domínio, principalmente, o perfil do domínio na ferramenta MVCASE e o gerador de código, são desenvolvidas mais facilmente e rapidamente diferentes aplicações do domínio.

Para validar a abordagem proposta foram desenvolvidos três estudos de caso no domínio da educação medica construtivista, validados no curso de medicina da Universidade Federal de São Carlos. Nesse domínio foram identificada quatro aplicações diferentes, e dessas foram implementadas três aplicações, o PRE, o MobilePRE e a P2PMobileLearning.

O apoio computacional desenvolvido no estudo de caso foi integrado na ferramenta MVCASE. A avaliação do desenvolvimento de um dos estudos de casos mostrou que a abordagem trouxe diversos ganhos no processo de desenvolvimento, principalmente no

tempo de desenvolvimento, na quantidade de artefatos reutilizados e na manutenção das aplicações.

As principais contribuições da abordagem para o desenvolvimento de software na Computação Ubíqua são:

a) Reúso: Através das ferramentas e artefatos construídos na Engenharia de Domínio, é possível desenvolver mais facilmente e rapidamente diversas aplicações para um domínio específico através do reúso de um framework do domínio e da geração de código para uma plataforma de implementação específica.

b) Produtividade: Através do Reúso e das ferramentas que apóiam os Engenheiros de Domínio e de Aplicação é possível atingir uma alta produtividade no desenvolvimento das aplicações do domínio.

c) Manutenção: Como boa parte da manutenção das aplicações é feita principalmente nos modelos, essa tarefa é facilitada, já que é mais fácil modificar os modelos do que o código fonte.

d) Facilidade na Modelagem dos Requisitos: Com modelos em um alto nível de abstração é mais fácil transformar os requisitos das aplicações em um modelo que possibilita a geração de código.

e) Geração de Testes Automatizados: São gerados testes automatizados dos requisitos do domínio, para frameworks como o JUnit e o Selenium, facilitando também as tarefas dos analistas de teste e ajudando a melhorar a qualidade das aplicações do domínio.

7.1. Dificuldades e Limitações

A principal dificuldade encontrada no desenvolvimento do trabalho foi a falta de documentação de alguns dos frameworks do Eclipse Modeling Project, o que dificultou a implementação das ferramentas do domínio.

As principais limitações da abordagem são referentes à complexidade no desenvolvimento dos artefatos reutilizáveis na Engenharia de Domínio. Desde a definição dos perfis, até a implementação do gerador de código. Alguns trabalhos estão sendo desenvolvidos para diminuir essas complexidades, tais como: transformação automática, com o framework Eclipse ATL, da ontologia do domínio para o perfil UML; e criação de *guidelines* para a implementação das regras de transformação entre modelos.

Outra limitação é que a abordagem só traz benefícios para domínios onde existem diversas aplicações diferentes, já que o esforço na Engenharia de Domínio é muito grande e para o desenvolvimento de poucas aplicações, esse esforço acaba não sendo recompensado.

7.2. Trabalhos Futuros

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

Na abordagem, trabalhos futuros incluem a criação de uma ferramenta totalmente integrada ao IDE Eclipse que auxiliará os Engenheiros de Software na execução de todas as atividades do processo proposto, a partir da descrição do domínio em uma ontologia até a definição dos testes automatizados, definir regras de transformação para automatizar a criação do perfil do domínio a partir da ontologia que descreve o domínio e estender a abordagem para utilizar o conceito de Linhas de Produto de Software [AHM08].

E finalmente, extender a abordagem com as disciplinas de Modelagem de Negócios, Gerenciamento de Configuração, Gerenciamento de Projeto e Ambiente. Essas disciplinas podem se beneficiar com o aumento de produtividade, melhor entendimento do impacto da ubiqüidade em relação aos negócios de software e melhor gerenciamento das modificações nos requisitos.

Em relação ao PRE e ao domínio utilizado, trabalhos futuros incluem estender o perfil do domínio para abranger uma maior quantidade de aplicações, incluindo também elementos para a gestão acadêmica do curso.

Também podem ser incluídos novos serviços para adaptação de conteúdo na P2PmobileLearning (e.g., vídeo e áudio), que na sua versão atual só adapta imagens e textos.

Referências

[AHM08] Ahmed, F., Capretz, L. F. 2008.: "The software product line architecture: An empirical investigation of key process activities". Journal Information Software Technology Volume 50, 1098-1113.

[ANT07] Agência Nacional de Telecomunicações, Brasil deve atingir 100 milhões de celulares ainda esse mês, 2007.

[ATL10] Disponível em http://www.eclipse.org/m2m/atl/, acessado em 05/03/2010.

[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.

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

[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.

[BUR08] Burghardt, C. Wurdel, M. Propp, S. Kirste, T. Forbrig, P.: "Tools support for intelligent environments". Anais da Intelligent Environments. pp. 1 – 4, 2008.

[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, 2007.

[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.

[EMP10] EMP - Eclipse Modeling Project, Disponível em http://www.eclipse.org/modeling/, Último acesso em junho de 2009.

[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

[GRO01] Grose, T. J., Doney, G. C., and Brodsky, S. A. “Mastering Xmi: Java Programming with Xmi, XML and UML”. John Wiley & Sons, Inc, 2001.

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

[HAR00] Harrison, W.; Ossher, H.; Tarr, P.: “Software Engineering Tools and Environment: A Roadmap”. In: The Future of Software Engineering. ACM, 2000. p. 261 -277, 2000.

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

[JAM09] Jamae, J. and Johnson, P. “Jboss in Action: Configuring the Jboss Application Server”. Manning Publications Co, 2009.

[JUN06] Jung, Y., Lee, J., and Kim, M. “Multi-agent based community computing system development with the model driven architecture”. In Proceedings of the Fifth

international Joint Conference on Autonomous Agents and Multiagent Systems. AAMAS '06. pp 1329-1331, 2006.

[KEL08] Kelly, S., Tolvanen, JP.: "Domain-Specific Modeling: Enabling full code generation", John Wiley & Sons, ISBN 9780047003666, 427 p, 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.

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

[KON07] Kongsli, V. 2007. "Security testing with Selenium.". Conference on Object-Oriented Programming Systems and Applications Companion (Montreal, Quebec, Canada, October 21 - 25, 2007.

[KUL07] Kulkarni, D. e Tripathi "Generative Programming Approach for Building Pervasive Computing Applications" Anais da International Conference on Software Engineering Workshop, pp. 189, 2007.

[LAF08] Laforcade, P., Zendagui, B., and Barré, V.. A Domain-Specific-Modeling Approach to Support Scenarios-Based Instructional Design. In Proceedings of the 3rd European Conference on Technology Enhanced Learning: Times of Convergence: Technologies Across Learning. P. Lecture Notes In Computer Science, vol. 5192. Springer-Verlag, Berlin, Heidelberg, 185-196, 2008.

[LYY02] Lyytinen, K., Yoo, Y.: "Issues and challenges in Ubiquitous Computing". Communications of ACM, vol. 45, no. 12, 2002.

[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.

[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 Janeiro de 2010.

[OMG08] OMG (2008). "Model Driven Architecture". Disponível em http://www.omg.org/mda/ , último acesso em abril de 2009.

[PAT06] Patel, S. N., Kientz, J.A., Hayes, G.R., Bhat, S., Abowd, G.D.: "Farther Than You May Think: An Empirical Investigation of the Proximity of Users to their Mobile Phones". Anais da Ubicomp, pp. 123-140, 2006.

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

[PRE06] Pressman, R. S. "Engenharia de Software" 6º Edição McGrawHill, 2006.

[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.

[SAN08] Santana, L. H. Z. “ADeSCoU: Uma Abordagem Para o Desenvolvimento de Software na Computação Ubíqua”. Dissertação de Mestrado, Departamento de Computação, Universidade Federal de São Carlos, São Carlos – SP, 2008.

[SAN09] Santana, E. F. Z., Oliveira, R. P., Prado, A.F., de Souza, W. L.: “Modelagem Específica de Domínio em Linhas de Produto de Software na Computação Ubíqua”. Simpósio Brasileiro de Componentes, Arquitetura e Reúso de Software, 2008.

[SAN09B] Santana, E. F. Z. ; Prado, A. F. ; Souza, W.L. . MMVCASE: Extensão da ferramenta MVCASE para suporte ao desenvolvimento de aplicações móveis. In: Simpósio Brasileiro de Sistemas Multimídia e Web (Webmedia), 2009.

[SAN09C] Santana, E. F. Z. ; Santana, L. H. Z. ; Prado, A. F. ; Souza, W.L. . “A Web 2.0 Based Environment for Simulation Activities in Constructivist Medical Education”. IADIS WWW/Internet Conference, 2009.

[SAN09D] Santana, E. F. Z. ; Prado, A. F. ; Souza, W.L. . P2PmobileLearning: Uma Aplicação P2P para compartilhamento de conteúdo em dispositivos móveis. In: Simpósio Brasileiro de Sistemas Multimídia e Web (Webmedia), 2009.

[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.: “Checklist to Characterize Ubiquitous Software Projects.” Anais do Sim, Brasileiro de Engenharia de Software, pp. 39-55, 2007.

[TEA00] Team, P. D. “PostgreSQL Programmer's Guide.” iUniverse, Incorporated, 2000.

[VAL09] Vale, S. Hammoudi, S. (2009): "COMODE: A Framework for the Development of Context-Aware Applications in the Context of MDE" In: Internet and Web Applications and Services, 2009.

[VAR08] Varga, C. R. R., Almeida, V. C., Germano, C. M. R., Melo, D. G., Chachá, S. G. F., Souto, B. G. A., Fontanella, B. J. B., Lima, V. V. . "O uso de simulações no processo ensino aprendizagem em medicina". Revista Brasileira de Educação Médica, 2008.

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

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

[WEI91] Weiser, M.: The Computer For The Twenty-First Century. In: Scientific American, vol. 265, no.3, pp. 94-104, 1991.

[WEI99] Weiser, M., Gold, R., Brown, J. S.: "The origins of ubiquitous computing research at PARC in the late 1980s". IBM System Journal, vol. 38, no. 4, pp. 693-696, 1999.

[WIL06] Willians, M.: "No Japão, internauta já navega mais pelo celular do que pelo computador". IDGNow, 2006.

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

Publicações

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

Santana, E. F. Z. ; Santana, L. H. Z. ; Prado, A. F. ; Souza, W.L. . A WEB 2.0 BASED ENVIRONMENT FOR SIMULATION ACTIVITIES IN CONSTRUCTIVIST MEDICAL EDUCATION. In: IADIS WWW/Internet Conference, 2009, Roma. Proceedings of IADIS WWW/Internet Conference, 2009.

Santana, E. F. Z. ; Santana, L. H. Z. ; Prado, A. F. ; Souza, W.L. . Um Ambiente Baseado na Web 2.0 para Atividades Simuladas na Educação Medica Construtivista. In: Simpósio Brasileiro de Informática na Educação (SBIE), 2009, Florianopolis - SC. Anais do Simpósio Brasileiro de Informática na Educação.

Santana, E. F. Z. ; Oliveira, R. P. ; Prado, A. F. ; Souza, W.L. ; Biajiz, M. . Modelagem Específica de Domínio em Linhas de Produto de Software na Computação Ubíqua. In: Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS), 2009, Natal. Anais do Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software,2009.

Santana, E. F. Z. ; Prado, A. F. ; Souza, W.L. . MMVCASE: Extensão da ferramenta MVCASE para suporte ao desenvolvimento de aplicações móveis. In: Simpósio Brasileiro de Sistemas Multimídia e Web (Webmedia), 2009, Fortaleza - CE. Anais do VIII Workshop de Ferramentas e Aplicações, 2009.

Santana, E. F. Z. ; Prado, A. F. ; Souza, W.L. . P2PMobileLearning: Uma aplicação P2P com adaptação de conteúdo em dispositivos móveis. In: Simpósio Brasileiro de Sistemas Multimídia e Web (Webmedia), 2009, Fortaleza - CE. Anais do Simpósio Brasileiro de Sistemas Multimídia e Web, 2009.

Santana, E. F. Z. ; Santana, L. H. Z. ; Prado, A. F. ; Souza, W.L. . Uma arquitetura P2P para adaptação de conteúdo em dispositivos móveis. In: Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia), 2008, Vila Velha - ES. Anais do Simpósio Brasileiro de Sistemas Multimídia e Web. Porto Alegre (RS), 2008. v. 2. p. 57-60.