ITALO RODRIGUES CASTRO

UMA PROPOSTA DE EXTENSÕES À UML PARA MODELAGEM DE APLICAÇÕES EM AMBIENTES MÓVEIS PARA APLICAÇÕES ORIENTADAS A SERVIÇOS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae.*

VIÇOSA
MINAS GERAIS - BRASIL
2008

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca Central da UFV

T
C355p
2008

Castro, Italo Rodrigues, 1981-
Uma proposta de extensões à UML para modelagem de aplicações em ambientes móveis para aplicações orientadas a serviços / Italo Rodrigues Castro. – Viçosa, MG, 2008.
xv, 70f. : il. ; 29cm.

Orientador: Mauro Nacif Rocha.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 67-70.

1. UML (Computação). 2. Computação móvel. 3. Métodos orientados à objetos (Computação). 4. Software - Desenvolvimento. I. Universidade Federal de Viçosa. II.Título.

CDD 22.ed. 005.1

ITALO RODRIGUES CASTRO

UMA PROPOSTA DE EXTENSÕES Á UML PARA MODELAGEM DE APLICAÇÕES EM AMBIENTES MÓVEIS PARA APLICAÇÕES ORIENTADAS A SERVIÇOS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 27 de fevereiro de 2008.

Prof. Alcione de Paiva Oliveira

Prof. Jugurta Lisboa Filho
(Co-orientador)

Profª Irís Fabiana de Barcelos Tronto

Prof. Vladimir Oliveira Di Iorio

Prof. Mauro Nacif Rocha
(Orientador)

Aos meus pais Ibsen Araújo da Silveira Castro

e Marlene Rodrigues Teixeira Castro

e ao meu orientador Mauro Nacif Rocha.

AGRADECIMENTOS

- A Deus por me dar força e sabedoria para vencer, conquistando mais este sonho.
- A minha família, em especial meus pais que sempre me deram apoio e incentivo quando precisei.
- Ao meu irmão, Thales Rodrigues por sempre me apoiar em todos os momentos.
- Ao Professor Dr. Mauro Nacif Rocha, orientador desta dissertação, pelo apoio e ajuda na conclusão deste trabalho.
- Aos co-orientadores José Luis Braga e Jugurta Lisboa Filho.
- Aos amigos Juliano Lino Ferreira e minha "irmãzinha" do mestrado, Daniella Inácio (Rosinha), pela amizade, companherismo e ajuda técnica sempre que precisei.
- Ao secretário da pós-graduação Altino Alves de Souza Filho, por estar sempre disposto a ajudar e pela sua amizade.
- À chefe de expediente Eliana Ferreira Rocha, pelo carinho e dedicação.
- A todos os professores e colegas do mestrado pelo convívio.
- À Universidade Federal de Viçosa.

BIOGRAFIA

Italo Rodrigues Castro, filho de Ibsen Araújo da Silveira Castro e Marlene Rodrigues Teixeira Castro, brasileiro nascido em 17 de maio de 1981 no município de Ipatinga, no estado de Minas Gerais.

No ano de 1999 após concluir o curso científico no Colégio Técnico de Coronel Fabriciano Padre de Man, iniciou o curso de graduação em Ciência da Computação nas Faculdades Integradas de Caratinga, onde obteve o grau de bacharel em Ciência da Computação no ano 2002. Em 2004 iniciou o curso de pós graduação *Lato Sensu* no Centro Universitário do Leste de Minas Gerais, onde teve sua primeira experiencia docente no curso de bacharelado em Sistemas de Informação, obtendo em 2006 o título de especialista em Tecnologia de Desenvolvimento de Software. No ano de 2005 começou a cursar o mestrado em Ciência da Computação na Universidade Federal de Viçosa - UFV.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

AGPS	*Assisted GPS*
AP	*Access Point*
API	*Application Program Interface*
CIS	*Context Information Service*
CORBA	*Common Object Request Broker Architecture*
DCOM	*Distributed component object model*
DoD	*Departament of Defense of U.S.A.*
ERB	Estação Rádio Base
EJB	*Enterprise JavaBeans*
GPS	*Global Positioning System*
HTML	*HyperText Markup Language*
IEEE	*Institute of Electrical and Electronics Engineers*
JEE	*Java Enterprise Edition*
JME	*Java Micro Edition*
JSE	*Java Standard Edition*
LBS	*Location Based Services*
LIS	*Location Interference Service*
MAC	*Medium Access Control*
MDA	*Model Driven Architecture*
MOF	*Meta Object Facility*

OO	Orientação à Objetos
OCL	*Object Constraint Language*
OMG	*Object Management Group*
OMT	*Object Modeling Technique*
OOSD	*Object Oriented System Development*
OOSE	*Object Oriented Software Engineering*
PDA	*Personal Digital Assistant*
PIM	*Plataform Independent Model*
RMI	*Remote Method Invocation*
SIG	Sistema de Informação Geográfica
SMS	*Short Message Service*
SLC	Servidor de Sistema de Localização
SBL	Serviços Baseados em Localização
SOA	*Service Oriented Architecture*
SOAP	*Service Oriented Application Protocol*
UM	Unidade Móvel
UML	*Unified Modeling Language*
XML	*Extensible Markup Language*
WSDL	*Web Service definition Language*
UDDI	*Universal Description Discovery Integration*
URL	*Uniform Resource Locator*
WLAN	*Wireless Local Area Network*

RESUMO

CASTRO, Italo Rodrigues, M.Sc.,Universidade Federal de Viçosa, fevereiro de 2008. **Uma Proposta de Extensões à UML para Modelagem de Aplicações em Ambientes Móveis para Aplicações Orientadas a Serviços**. Orientador: Mauro Nacif Rocha. Co-orientadores: José Luis Braga e Jugurta Lisboa Filho.

Com o grande crescimento do uso de dispositivos móveis (*laptop, palmtop*, ou algum tipo *Personal Digital Assistants* (PDA)), surgiu um novo paradigma da computação: a computação móvel. O novo paradigma permite que os usuários desse ambiente tenham acesso a serviços independente de sua localização, ou seja, permite a mobilidade. Estão sendo feitos vários estudos na área de computação móvel para atender de forma cada vez melhor às necessidades dos usuários. O desenvolvimento de *softwares* nesta área está aumentando e por esta razão os desenvolvedores desses tipos de sistemas estão adotando estratégias para aumentar a qualidade e a produtividade de *softwares* de serviços para ambientes móveis. Para atingir esta finalidade, foi feito um estudo da *Service Oriented Architecture*(SOA) para otimizar o tempo gasto no desenvolvimento desses tipos de sistemas. A SOA é uma abordagem de arquitetura a partir de serviços autônomos. A *Unified Modeling Language* (UML) é uma linguagem de modelagem padrão para *softwares* e sistemas orientados a objetos. Ela foi projetada para visualizar, especificar, construir e documentar vários aspectos de um sistema. Com o intuito de minimizar os problemas existentes no desenvolvimento desses sistemas, foi proposto neste trabalho um perfil UML, constituído por um grupo predefinido de estereótipos e restrições que em conjunto especializam e configuram a UML para a modelagem de serviços para ambientes móveis. O perfil proposto foi

avaliado através da modelagem de uma aplicação real baseada na SOA através de diagramas UML.

ABSTRACT

CASTRO, Italo Rodrigues, M.Sc.,Universidade Federal de Viçosa, february, 2008. **A Proposal of UML Extensions for Modeling of Applications in Mobile Environments for Service Oriented Applications**. Advisor: Mauro Nacif Rocha. Co-advisors: José Luis Braga and Jugurta Lisboa Filho.

With the great growth of the use of mobile devices (laptop, palmtop, or some type Digital Personal Assistants (PDA)), a new paradigm computing appeared: the mobile computing. The new paradigm allows the users of that environment to have access to services independent of their location, in other words, it allows the mobility. They are being made several studies in the area of mobile computing to assist well and better in way to the necessities of users. The development of softwares in this area is increasing extraordinarily. For this reason the architects of those types of systems are adopting strategies to get better and to increase the productivity of softwares of services for mobile environment. To reach this purpose, it was studied the Service Oriented Architecture(SOA) to optimize the time necessary in the development of those type of systems. The SOA is an architecture approach for the generation of created systems starting from autonomous services. The Unified Modeling Language (UML) is a language of modeling pattern for softwares and systems objects oriented. It was projected to visualize, specify, build and document several aspects of a system. With the intention of minimizing the existent problems in the development of those systems was proposed in this work a UML profile, constituted by a group previously defined of stereotypes and restrictions that together specialize and they configure UML for the modeling of services for mobile environment. The proposed profile was

evaluated through the modeling of a real application based in SOA through UML diagrams.

Capítulo 1

Introdução

Atualmente o uso de dispositivos móveis está crescendo de forma extraordinária. Estima-se que daqui a poucos anos dezenas de milhões de pessoas terão um *laptop, palmtop* ou algum tipo *Personal Digital Assitants* (PDA). A computação móvel representa um novo paradigma computacional que surgiu logo após as redes de computadores na década de oitenta. A comunicação sem fio elimina a necessidade do usuário manter-se conectado a uma estrutura fixa. A computação móvel permite que os usuários desse ambiente tenham acesso a serviços independente de sua localização, ou seja, permite a mobilidade (MATEUS, 2004).

De acordo com Mateus(2004), a comunicação entre dispositivos móveis com outro elemento computacional é feita através de um canal de comunicação sem fio. A mobilidade tornou-se um novo paradigma em sistemas distribuídos que está causando mudanças significativas na forma de projetar e desenvolver sistemas de *software*. A combinação de comunicação sem fio com a mobilidade tem exercido uma forte influência na pesquisa e no desenvolvimento em várias áreas da computação e telecomunicações, em especial em redes de computadores, sistemas operacionais, otimização, sistemas de informação, banco de dados, engenharia de *software*, entre outras.

O aumento da mobilidade tem gerado a necessidade de novas aplicações que permitam o acesso a informações de forma eficiente no ambiente de comunicação sem fio, surgindo a necessidade de modelar serviços para este tipo de ambiente. Exis-

tem vários tipos de serviços para o ambiente móvel, como por exemplo serviços de localização, m-commerce, entre outros.

O trabalho aborda os serviços comuns para usuário móvel e baseados na sua localização, levando em conta as restrições temporais impostas pela mobilidade. Os Serviços Baseados em Localização (SBL) são definidos como serviços que fornecem informações ao usuário dependendo de sua localização geográfica (SCHILLER, 2004). Esses serviços têm recebido uma atenção especial por vários pesquisadores sendo muito utilizados atualmente (SCHILLER, 2004).

Além disso, devem ser levadas em consideração questões como consumo de energia, riscos de perda ou extravio de dados, variações bruscas na latência e na largura de banda, redes heterogêneas, segurança, relevância das informações, tempo de resposta entre outras, para que este estudo resulte em tecnologias e produtos que sejam utilizados e aceitos pelo mercado e que atendam de forma cada vez melhor às necessidades dos usuários.

Para a modelagem desses serviços, um recurso muito utilizado é a *Unified Modeling Language* (UML), uma linguagem de modelagem visual orientada a objetos. As linguagens de modelagem orientadas a objetos surgiram entre a metade da década de 1970 e o final da década de 1980. Os desenvolvedores começaram a experimentar outros métodos de análise e projeto a partir das experiências adquiridas com a programação orientada a objetos. Atualmente a UML vem se tornando um padrão para o desenvolvimento orientado a objetos, tanto para o mercado como para o meio acadêmico (FOWLER, 2003).

1.1 O problema e sua Importância

Tradicionalmente, a indústria de *software* tem apresentado grande investimento nas atividades ligadas à produção de *softwares* como programação, testes, integração de componentes e de sistema. Isso se opõe às atividades ligadas ao projeto e à concepção, tais como a engenharia de requisitos, análise e o projeto. A ênfase deve ser reforçada nas atividades de concepção e projeto, e consequentemente, o esforço nas

atividades de produção deve ser minimizado e realizado tanto quanto possível de forma automática, tal como acontece em outras indústrias como, por exemplo, na indústria automobilística, alimentar ou farmacêutica (PAULA FILHO, 2003).

Os aspectos relacionados com o "como fazer" são relevantes, mas são tratados principalmente pelos engenheiros de *software* que são os responsáveis por providenciar *softwares* de qualidade, flexíveis e reutilizáveis, muitas vezes o produto não possui preços e prazos facilmente previsíveis e devidamente controlados (PAULA FILHO, 2003).

Este trabalho pretende explorar técnicas e mecanismos que permitam acelerar e melhorar a atividade de gestão e de produção de *software* para sistemas móveis. Para contribuir com este objetivo será definido e proposto um conjunto integrado de elementos que consiste numa abordagem para desenvolvimento de *software* fortemente baseada em modelagem, utilizando a UML (ERIKSON, 2004).

Devido ao fato da UML não ter primitivas específicas que permitam a modelagem de serviços em sistemas móveis, existem muitos projetos feitos hoje para aplicações destinadas a operar em ambientes de computação móvel, com recursos computacionais e interfaces restritas, que ignoram os diversos problemas existentes nesse tipo de ambiente, como limitação de banda, interferências, devido à mobilidade. Estes aspectos diferem dos ambientes computacionais e de comunicação tradicionais, constituídos de computadores pessoais ou *workstations* e redes cabeadas. Embora a UML não seja diretamente responsável por tais problemas, pode levar os projetistas e desenvolvedores a não levar em conta os problemas relacionados a esses ambientes. Alguns desses problemas são a perda de requisitos, problemas com poucos detalhes, levando a soluções que não atendem aos requisitos e já entram em produção necessitando alterações.

Com o objetivo de minimizar esses problemas este trabalho foi desenvolvido utilizando a *Service Oriented Architecture* (SOA), que é uma arquitetura de desenvolvimento de *software* baseada em componentes de negócio ou serviços reutilizáveis (MCCOY, 2007).

Esses serviços são desenvolvidos em diferentes linguagens de programação, hos-

pedadas em plataformas diferentes, com uma variedade de modelos de segurança e regras de negócio. Os Serviços baseados em SOA são módulos de negócios ou funcionalidades de aplicação com interfaces bem definidas e normalmente invocadas com mensagens.

Segundo o Instituto Gartner, a adoção da arquitetura orientada a serviços está crescendo com velocidade dramática, sendo utilizado em mais de 50% das novas aplicações operacionais de missão crítica e processos de negócios em 2007. O instituto garante que em 2010, 80% das novas aplicações serão baseadas na SOA, tornando-a prática predominante de Engenharia de *Software*, direcionando a uma plataforma unificada de negócios (MCCOY, 2007).

Com a crescente complexidade dos Sistemas de Informação, está mudando a maneira dos engenheiros e arquitetos de *software* trabalharem. Eles estão focando sua atenção nos aspectos de modelagem e no processo de desenvolvimento de sistemas, sem se preocupar com detalhes de implementação. Modelos proporcionam um maior nível de abstração, permitindo trabalhar com sistemas maiores e mais complexos de uma maneira simples, facilitando a implementação do sistema em diversas plataformas.

Um modelo é a descrição de um sistema, descrito em uma linguagem bem definida, ou seja, com sintaxe e semântica precisa e que pode ser interpretado e manipulado por um computador (PAULA FILHO, 2003). Entre as linguagens de modelagens definidas pela OMG (*Object Management Group*), a mais usada é a UML.

Entretanto, existem situações em que a UML não é apropriada para modelar aplicações de um domínio específico. Isso acontece quando a sintaxe e a semântica da UML não permite expressar os conceitos específicos do domínio, ou quando se deseja restringir e especializar os elementos da UML que são genéricos e numerosos.

Os criadores da UML 2.0 propuseram o conceito de "Perfil", que permite a modelagem de artefatos utilizando a UML de acordo com as necessidades específicas em cima das tecnologias (ERIKSON, 2004). É constituído por um grupo predefinido de estereótipos e restrições, que em conjunto, especializam e configuram a UML para um dado tipo de aplicação ou para um determinado processo de desenvolvimento. Por exemplo "JAVA *Profile*", é definido especificamente para modelar aplicações desen-

volvidas na linguagem JAVA (FOWLER, 2003).

Diante disso, o presente trabalho tem como principal objetivo mostrar a importância da modelagem de *software* de serviços para ambientes móveis, verificar a aplicabilidade da UML para a modelagem desses serviços baseados em SOA e propôr um conjunto de extensões UML, constituindo um Perfil UML para este domínio, permitindo a modelagem desses sistemas de acordo com as restrições impostas pela mobilidade, como características do ambiente devido a mobilidade, tempo, características pessoais e culturais, entre outras. Assim, será possível prevenir e tratar os problemas descritos anteriormente, diminuindo o custo.

O trabalho foi avaliado e testado através de uma aplicação-exemplo baseada em SOA, onde foram utilizadas as extensões UML propostas para adequar os métodos, técnicas e padrões de desenvolvimento desses tipos de sistemas.

1.2 Objetivos do Trabalho

Esta pesquisa tem como objetivo geral propor um conjunto de extensões UML para modelar serviços para aplicações em ambientes móveis, permitindo a especificação visual de alto nível desse tipo de sistema. Especificamente, os objetivos detalhados são:

- Levantamento bibliográfico no contexto de Computação Móvel, serviços e tecnologias envolvidas;
- Levantamento bibliográfico sobre UML e as possíveis extensões existentes, obtendo uma visão crítica para as mesmas;
- Proposição de um conjunto de extensões UML para modelagem de serviços para aplicações em ambientes móveis, constituindo um Perfil para esse domínio;
- Fazer a avaliação e testes do Perfil utilizando uma aplicação real.

1.3 Trabalhos Relacionados

Foi feita uma cuidadosa busca nos principais sites de busca (google, google scholar, acm, periodicos.capes.gov.br, etc.), em buscas de artigos científicos, além de livros e documentação da UML. Foi encontrado alguns trabalhos semelhantes a este, mas não um específico que propõe extensões UML para modelagem de serviços em ambientes móveis baseados na SOA. Foram encontrados vários trabalhos que propõem um Perfil UML para diversos tipos de sistemas, tais como Sistemas de Tempo Real, Sistemas Web, citados em: (GRAF, 2005), (CONALLEN, 1999), respectivamente.

Um outro tipo de trabalho foi encontrado citado em (GILMORE, 2006), onde são abordadas as ferramentas de *software* para análise quantitativa de projetos de aplicações móveis, onde foi feito a modelagem da mobilidade utilizando a UML através de Diagramas de Atividades.

O Perfil para Sistemas de Tempo Real proposto por GRAF (2005), preocupa-se com o comportamento do sistema em relação ao tempo, sendo este um fator principal desses sistemas, em que qualquer falha poderá trazer grandes prejuízos. Já os sistemas para web são os que mais se assemelham aos sistemas móveis, seguindo a arquitetura Cliente-Servidor, semelhante a requisição e prestação de serviços. Porém, o Perfil proposto não resolve os problemas existentes no ambiente móvel, como por exemplo, prestar serviços para um usuário móvel que requisita a localização de um hospital mais próximo dele. Isso implica em vários problemas a serem resolvidos, sendo que um deles é o tempo que a informação será útil, devido ao fato do usuário estar em movimento.

Durante o refinamento da pesquisa foram encontrados alguns trabalhos relacionados com Perfil para Sistemas Móveis, Sistemas Distribuídos Móveis e modelagem para SMS *(Simple Mobile Services)* citados em (GRASSI, 2004), (SIMONS, 2007) e (BROLL, 2007), respectivamente. O Perfil para sistemas móveis se preocupa com a modelagem de uma entidade móvel, que entidades são móveis e quais os fatores que causam o movimento delas. O segundo, se preocupa em preservar a privacidade dos usuários no ambiente distribuído utilizando tecnologias como CORBA, DCOM, etc.

O terceiro, faz a modelagem de SMS que facilita o desenvolvimento no contexto de serviços móveis.

Por fim, foram encontrados alguns trabalhos sobre Perfis UML para Arquiteturas Orientadas a Serviços e Web Services, citados em:(HECKEL, 2003), (ORTIZ, 2006), respectivamente. Os dois Perfis UML tratam de questões como interoperabilidade entre as aplicações, reusabilidade e plataformas independentes. O primeiro define um perfil UML para a SOA, representando o cliente que requisita serviços, o provedor de serviços e o registro de serviços, que possui quais interfaces/serviços são implementados para serem acessados posteriormente. Esta arquitetura é descrita de forma mais detalhada no capítulo 2.3. Estas propostas de Perfis UML não levam em consideração a prática de serviços em ambientes móveis, como as restrições impostas pela mobilidade, por isso não foram usadas neste trabalho.

1.4 Metodologia

O aumento da complexidade dos sitemas e da utilização de ferramentas de desenvolvimento de sistemas orientados a objetos, tornou-se necessário o uso de uma metodologia unificada de desenvolvimento. Com o uso dessa metodologia foi possível facilitar a compreensão, implementação e testes desses sistemas (SOMMERVILLE, 2007).

Assim, com este interesse, foi desenvolvido neste projeto um conjunto de extensões UML para modelar serviços em ambientes móveis, fazendo sua avaliação e testes em uma aplicação real. Para isto, foi feito o levantamento bibliográfico no contexto de Computação Móvel, Serviços Baseados em Localização (LBS) e tecnologias envolvidas, e também foi realizado uma revisão de literatura sobre a SOA, sendo visto as vantagens, desvantagens e peculiaridades desta arquitetura. Também foi feito o estudo bibliográfico de artigos e documentações especializadas em UML e das possíveis extensões existentes, bem como de livros dos principais autores e dos criadores da UML.

Após a revisão de literatura, foi definido o perfil *Mobile Services* para modelagem de serviços em ambientes móveis, de acordo com as necessidades de produção

de serviços para este tipo de ambiente, juntamente com os conceitos necessários para esse propósito.

Finalmente, foi apresentada uma aplicação real descrita no capítulo 4, onde foi feita a análise e projeto do modelo proposto utilizando diagramas UML para avaliar o Perfil *Mobile Services* proposto.

1.5 Organização deste Documento

- No Capítulo 2, são apresentados os principais conceitos abordados dentro do contexto do desenvolvimento deste trabalho.
- No Capítulo 3, é apresentado o Perfil UML proposto, justificando cada extensão proposta para este Perfil.
- No Capítulo 4, é apresentado uma avaliação do Perfil UML proposto através de um exemplo de aplicação.
- No Capítulo 5, são descritos e interpretados os resultados.
- No Capítulo 6, são apresentadas as conclusões e algumas possíveis relações de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

2.1 Computação Móvel

> A Computação Móvel é uma área que visa criar soluções de negócios usando computadores e comunicações para permitir aos usuários trabalharem fora dos ambientes fixos, onde normalmente operam. O ambiente móvel ou ambiente da computação móvel baseia-se na capacidade que os usuários têm de, munidos de um dispositivo móvel (*Laptops, Notebook, PDA's*, celulares, etc), terem acesso a parte fixa da rede, e possivelmente com outros dispositivos móveis, independentemente da sua localização física (MATEUS, 2004, p.1).
>
> Ela representa um novo paradigma computacional que tem como objetivo prover ao usuário acesso permanente a uma rede fixa ou móvel independente de sua localização física, ou seja, é a capacidade de acessar informações, aplicações e serviços a qualquer lugar e a qualquer momento. A Computação móvel também recebe o nome de Computação ubíqua ou Computação nômade (MATEUS, 2004, p.1).

O dispositivo móvel deve ter poder de processamento, possibilitar a troca de informações na rede e ser compacto para que o usuário possa transportar. Para isso,

geralmente o dispositivo móvel possui tamanho reduzido e sem presença de cabos para a conexão com a rede ou com a bateria. (AMJAD, 2004).

Um diferencial importante da Computação Móvel é a interação entre ela e as diversas áreas da Ciência da Computação. Ela não trata somente de questões relativas às áreas de Sistemas Distribuídos e Redes de Computadores, na verdade, é um paradigma que trata de praticamente todas as áreas da Ciência da Computação. A área de Computação Móvel possui uma natureza multidisciplinar que caracteriza os sistemas móveis e ubíquos, e promove uma interação entre áreas de investigação diversas como, por exemplo, Sistemas Distribuídos, Inteligência Artificial, Eletrônica, Engenharia de *Software*, Computação Gráfica, Sistemas de Informação, Banco de Dados, entre outras.

Algumas das principais linhas de interesse dentro da Computação móvel são:

- Arquiteturas, protocolos e algoritmos contendo mobilidade, limitação de largura de banda, limitação de energia e/ou de conectividade intermitente.
- Aplicações e serviços para usuários móveis.
- Aspectos fundamentais de Computação móvel e redes sem fio.
- Aplicações de dependência de localização e protocolos.
- Modelagem, aspectos de medição e simulação de redes móveis.
- Sistemas de operação e suporte de middleware para computação e redes.
- Performance de mobilidade e redes sem fio e sistemas.
- Segurança, privacidade, e tolerância a falhas de sistemas móveis e sem fio.

2.1.1 Principais Problemas e Desafios

A mobilidade introduz problemas e desafios que não se via, ou eram ignorados em ambientes fixos. Há uma mudança de visão e pensamento que devem ser considerados. Alguns problemas que já foram resolvidos na computação fixa, ainda existem na computação móvel. Existem muitos problemas na computação móvel devido as limitações

impostas pela mobilidade como a velocidade do canal, interferências, localização do dispositivo móvel e a duração da bateria (AMJAD, 2004).

Estes e muitos outros parâmetros são fundamentais na hora de se projetar algoritmos para ambientes móveis, em que algumas vezes não se preocupava ao criar algoritmos baseados em redes fixas. Os projetos de *hardware* e *software* feitos para instalação e expansão dos sistemas de comunicação móvel quase sempre requerem grandes investimentos, o que tornam os problemas grandes desafios a serem resolvidos (AMJAD, 2004).

Hoje em dia, nos deparamos com vários problemas e desafios como (ZENI, 2006):

- **Características do Ambiente** - Capacidade de comunicação limitada com largura de banda variável e alta taxa de erros. Perda de dados, devido à falha do *software* ou extravio do equipamento;
- **Energia** - Energia limitada por baterias com limite de consumo, de forma que deve-se despender o mínimo de energia para o processamento do sistema;
- **Interface dos Dispositivos Móveis** - Possuem em geral telas pequenas e não possuem teclados e mouses comuns. Dessa maneira, aumenta a limitação na interação com o dispositivo;
- **Gerência de dados** - quando o usuário se desconecta e reconecta com a rede fixa, as modificações que foram feitas em arquivos durante o modo desconectado devem ser enviadas para o servidor apropriado;
- **Capacidade dos Dispositivos Móveis** - Esses dispositivos possuem limites físicos de hardware, devido a sua portabilidade. Também possuem poder de processamento e dispositivos limitados;
- **Heterogeneidade** - Em ambientes externos a velocidade de comunicação do dispositivo com a rede fixa, em geral é mais baixa que em ambientes internos que possuem uma melhor conectividade ao dispositivo móvel;

- **Necessidade de Adaptação** - no desenvolvimento de soluções devem ser levadas em consideração:

  Características físicas: - algumas características como energia do dispositivo (baterias), características do ambiente como montanhas, chuvas, etc., mobilidade, limitação dos dispositivos etc.;

  Características pessoais, culturais e lógicas: - características de acordo com o perfil de cada usuário, informações sobre localização etc. Cada dispositivo móvel geralmente são de cada usuário, dessa forma deve ser levado em consideração as preferências de cada usuário e do ambiente em que ele se encontra. Por exemplo, um serviço de informações sobre os hospitais mais próximos da localização do usuário;

- ***Seamless Communication*** - para que os dispositivos móveis não percebam que estão utilizando várias infra-estruturas diferentes de comunicação. Por exemplo, quando o usuário utilizando uma rede de comunicação celular sai de uma estação base para outra.

Devido a esses problemas, vários avanços já foram conquistados como por exemplo, desenvolvimento de novas arquiteturas de *hardware* que consumam pouca energia, desenvolvimento de arquiteturas de *software* que sejam flexíveis para fins de adaptação aos recursos limitados dos dispositivos móveis, novos conceitos e formas de interface com usuário, desenvolvimento de novas aplicações e serviços que atendam as necessidades da computação móvel.

2.1.2 Principais Aplicações e Serviços

Devido a mobilidade, é encontrado as maiores oportunidades de desenvolvimento de novos produtos comerciais que impulsionam cada vez mais o crescimento da computação móvel e de suas tecnologias. São mostradas a seguir alguns exemplos de aplicações e serviços para algumas áreas específicas (AMJAD, 2004):

- **Aplicações pessoais** - ex: agenda, acesso à informação, transferência de arquivos, web, email, dentre outras;

- **Aplicações Corporativas** - ex: acesso remoto, vendas, distribuição, transportes, estoque, automação industrial, dentre outras;
- **Aplicações Financeiras** - ex: transações bancárias, dentre outras;
- **Policiamento e Segurança** - ex: consulta de roubo, dados criminais, informações sobre boletim de ocorrências, dentre outras;
- **Entretenimento** - ex. jogos pessoais e interativos.
- **Serviços de informação em geral** - ex: trânsito, tempo, informações em geral, dentre outras;

  Na área de serviços, cita-se duas atrações comerciais para a área da computação móvel: Serviços de Comércio Eletrônico Móvel (*M-Commerce*) e Serviços Baseados em Localização (SBL).

- **Serviços de Comércio Eletrônico Móvel (*M-Commerce*)**- Área da computação móvel que deve ser levado em consideração características do ambiente, energia, personalização, possibilidades de interrupções na comunicação, perfis de usuários, dentre outras. Por exemplo, um usuário que está dentro de um *shopping*, e deseja receber informações sobre produtos de seu interesse. Assim, o usuário pode conferir o melhor produto, o melhor preço, condições de pagamento, realizando toda a transação eletronicamente (TSALGATIDOU, 2004).

Os Serviços Baseados em Localização (SBL) e os conceitos básicos e a *Service Oriented Architecture* (SOA), serão apresentados de forma mais detalhada nas sessões subsequentes.

2.2 Serviços Baseados em Localização (SBL)

"Serviços Baseados em Localização (SBL) são serviços de informações de localização oferecidos em um ambiente sem fio que dependem da localização do dispositivo móvel"(SCHILLER, 2004).

Estes serviços começaram sua existência na década de 70. O Departamento de Defesa dos EUA já operava o Sistema de Posicionamento Global (GPS), que informa a posição de pessoas ou objetos através de satélite. GPS é uma rede de satélites que informa a posição, em coordenadas geográficas, de um ponto qualquer da superfície terrestre. Com o uso do GPS, surgiram várias aplicações que usavam a informação de localização de um indivíduo para lhe prover serviços personalizados. Inicialmente, ele foi usado para propósitos militares. A partir da década de 80 o GPS passou a ser utilizado por indústrias do mundo todo. Entretanto, os serviços baseados em localização tiveram um grande salto a partir da década de 90 com o surgimento das redes móveis e os serviços de telefonia móvel, que proveram significantes operações móveis (SCHILLER, 2004).

A partir da década de 90, com o desenvolvimento de novas tecnologias de localização aumentou o desenvovimento desses serviços baseados na localização do usuário e também nos serviços baseados no perfil do usuário. Dessa maneira, as empresas passam a oferecer esses serviços de forma que atendam cada vez melhor os usuários desses serviços.

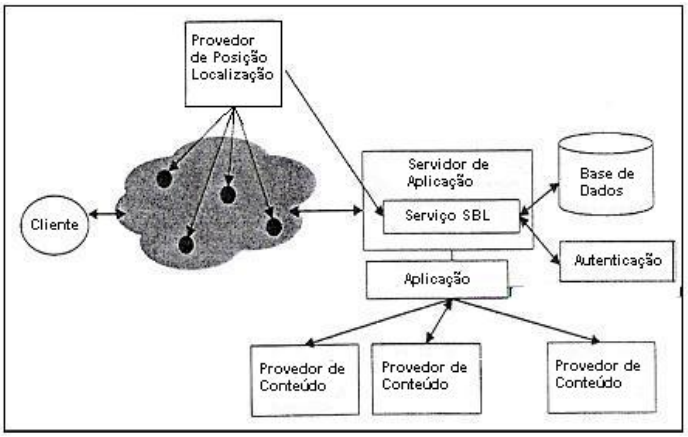
2.2.1 Arquitetura de Serviços Baseados em Localização (SBL)

Cada aplicação é processada em um Servidor de Aplicação que irá buscar em Provedores de Conteúdo as informações necessárias para informar ao cliente. Por exemplo, informar os dados e a localização de lojas, restaurantes, etc.

A Figura 2.1 mostra a arquitetura de SBL, onde o Servidor de Aplicação possui o serviço de SBL, devido ao fato da informação ser dependente da localização do cliente. Este serviço possui uma interface de autenticação que verifica em um banco de dados de assinantes se o cliente que está tentando acessar o serviço é ou não autorizado.

A localização ou posição do cliente é fornecida pelo Provedor de Posição-Localização, que irá conectar-se ao cliente através de terminais móveis ou pela *Internet*, podendo assim fornecer a localização do cliente para o Servidor de Aplicação. Depois que o Servidor de Aplicação processar a consulta, o resultado é enviado para

o cliente através de terminais móveis ou pela *Internet* (SCHILLER, 2004).



Fonte: Adaptado de SCHILLER (2004)

Figura 2.1: Arquitetura de SBL

2.2.2 Principais Aplicações de SBL

Um SBL pode ser usado para prover rotas de veículos, informações sobre condições de tráfego, lugares próximos como restaurantes e postos de gasolina (SCHILLER, 2004). Destaca-se as principais aplicações destes serviços como:

Serviços de informação: onde é utilizado um banco de dados para informar sobre a posição do usuário, como por exemplo, localização baseada em páginas amarelas e condição de tráfego em tempo real.

Serviços de monitoramento de veículos: para o caso de precisar de assistência ou em caso de roubo.

Serviços de gerenciamento de recursos: Estes serviços envolvem grande volume de dados e processamento. É utilizado, por exemplo, para informar qual a melhor loja de uma empresa de entregas.

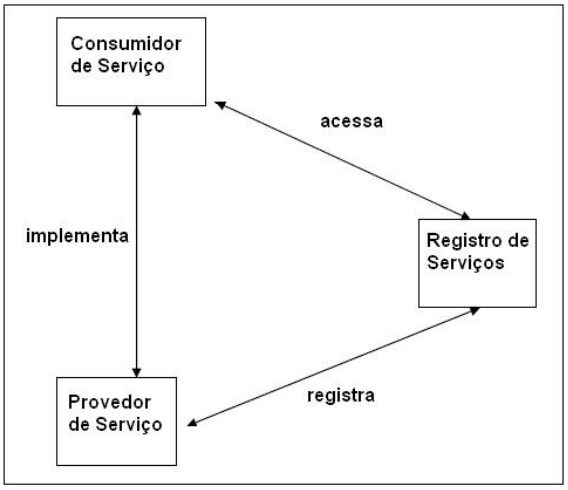
Serviços de rota: informa qual o melhor caminho para veículos ou pedestres mover de um ponto a outro.

2.3 *Service Oriented Architecture*(SOA)

2.3.1 Introdução

"Um serviço é uma função do sistema que pode ser facilmente vinculado a outros componentes de *software*. O acesso a esses serviços é realizado por meio de uma interface com as políticas e restrições específicas de cada serviço proposto"(OASIS, 2007). A Figura 2.2 mostra o serviço, que por sua vez, possui uma entidade que provê serviços chamado de provedor do serviço, pela entidade que utiliza os serviços chamado de consumidor do serviço e pelo registro de serviços.

O consumidor de serviços deseja acessar um serviço, ou seja uma interface de serviço que será implementada para satisfazer suas necessidades. Para isso, ele acessa o registro de serviços que possui a identificação do serviço fornecendo esta identificação para o provedor de serviços que implementa a interface que foi solicitada com os resultados do serviço para o consumidor de serviço. O Registro de serviços é onde ficam registrados os serviços que estão sendo processados. Além disso, possui também as interfaces que são implementadas pelos serviços para serem acessadas pelos usuários (STREET, 2008).



Fonte: Adaptado de (STREET, 2008)

Figura 2.2: Princípio da SOA

A SOA, é uma arquitetura de *software* baseada em componentes de serviços que

podem ser reutilizados (STREET, 2008). Cada componente possui uma ação específica, através de uma interface de serviço comopor exemplo, validar usuário (MCCOY, 2007).

A SOA possui uma nova visão aos conceitos de desenvolvimento orientado a objetos e baseado em componentes, onde as áreas de negócio, que possuem vários serviços, sejam independente de tecnologia, ou seja interoperabilidade. Dessa forma, os usuários poderão integrar tecnologias diferentes, por exemplo, utilizar, padrões *Java JEE, Corba, DCOM*, para aumentar o reuso de serviços existentes e a produtividade. Portanto, uma vez montada essa arquitetura não é mais necessário construir novas aplicações desde o início, que passam a ser montadas a partir das já existentes (MCCOY, 2007).

Um exemplo de tecnologia baseada em SOA são os *Web Services* ORTIZ (2006). Os *Web Services* foram criados para construção de aplicações para a Internet. As interfaces gráficas para os usuários do sistema são criados por outras empresas ou pessoas. *Web Services* utilisam a SOA usando protocolos de comunicação padronizados (SOAP, WSDL e UDDI) que descrevem dados através da linguagem XML (ALONSO, 2004).

O objetivo principal da SOA é o agrupamento de componentes para realizar algum serviço e ainda invocar outros serviços disponíveis, permitindo o reuso. Isso reduz o custo e o risco para o desenvolvimento desses componentes de serviços.

2.3.2 Características presentes na SOA

- **Caixa Preta**

  Os serviços interagem entre si por meio de provedores e consumidores de serviços. Eles são considerados uma caixa preta pelo fato de sua implementação ser encapsulada, não sendo visível pelo consumidor do serviço. Assim ele sabe somente qual serviço é oferecido e não como ele é implementado. Isto faz da SOA uma arquitetura ideal para ser utilizada em um ambiente computacional que possua *hardware* e *software* de múltiplos fabricantes (MACHADO, 2004).

- **Granularidade alta**

  A granularidade do artefato de *software* utilizado na arquitetura SOA é alto, pois aumenta a produtividade no desenvolvimento de *software*. Quanto maior a granularidade, maior desempenho e reuso. Componentes promovem reusabilidade, enquanto reduzem riscos e custos. Quando uma classe é reutilizada, um conjunto de rotinas é reutilizado (métodos), bem como uma estrutura de dados (atributos).

- **Ambiente Heterogêneo**

  Os sistemas baseados na arquitetura SOA são sistemas abertos ou sistemas abertos e distribuídos. O primeiro é quando são usadas arquiteturas já prontas em que as entidades de um sistema podem ser compostas dinâmicamente, e a segunda, além de usar arquiteturas prontas, podem estar localizadas em máquinas diferentes. Um exemplo é a arquitetura aberta de computadores, várias empresas podem desenvolver produtos utilizando a arquitetura já existente e propor melhorias de forma mais rápida, diminuindo o custo.

- **Composição**

  A SOA utiliza componentes que se comunicam entre si por troca de mensagens contendo dados. A Composição integra os componentes de serviços no sistema para realização de tarefas. Podemos visualizar a utilização da composição na linguagem Java, que carrega em tempo de execução novos componentes e invoca métodos específicos de uma interface definida anteriormente.

- **Robustez de Protocolos**

  Qualquer tipo de comunicação entre partes necessita de um protocolo. Porém, em ambientes distribuídos (principalmente se forem heterogêneos), onde as partes estão conectadas de uma rede de comunicação, os protocolos são mais relevantes. Além disso, devem ser levadas em consideração questões como desempenho, robustez e segurança são muito importantes.

- **Fracamente Interligados**

  Os serviços são inpendentes da implementação, plataforma e são fracamente interligados, ou seja podem ser utilizados por outros componentes de serviços ou em outros projetos. Dessa maneira, essas características permitem o reuso e a interoperabilidade dos mesmos, que são as principais motivações na utilização de desenvolvimento baseado em componentes. Isso implica no reaproveitamento de partes já desenvolvidas, reduzindo o tempo de desenvolvimento e o aumento da qualidade do produto final, o que é bastante atraente para empresas de tecnologias. A utilização de serviços de outros componentes é feita através de interfaces que conectam o componente de serviço em um ambiente e substituí em outro que execute o mesmo serviço, sem alterar o sistema.

2.3.3 Alguns Benefícios da SOA

A SOA traz diversos benefícios significativos, alguns são enumerados a seguir MACHADO (2004):

- **Investimento Inicial:** No início é necessário grande investimento em *hardware* e *software* para o desenvolvimento que utiliza a SOA. Um serviço é um conjunto de componentes existentes e são acessados por uma interface. Esses componentes de serviços podem estar em diferentes máquinas, sistemas operacionais e linguagens de programação.

- **Redução de tempo e custo de desenvolvimento:** Os novos serviços são desenvolvidos de forma mais rápida, devido ao reuso de componentes existentes. Dessa maneira, é possível adicionar o componente para o novo serviço e continuar trabalhando acrescentando os novos requisitos.

- **Redução de Riscos:** Os componentes existentes diminui a introdução de novas falhas na criação de um serviço. Quando se reutiliza um componente que esteja funcionando corretamente, a probabilidade é alta que ele continuará funcionando após o reuso.

2.3.4 Comparativo entre abordagens OO e SOA

A tabela 2.1 faz um comparativo entre a abordagem Orientada a Objetos (OO) e a abordagem Orientada a Serviços (SOA). A SOA em geral utiliza a abordagem OO, aumentando a produtividade da implementação dos serviços.

Tabela 2.1: Comparativo entre as abordagens OO e SOA

Conceito	OO	SOA
Unidade de desenvolvimento	Classes	Serviços
Foco principal	Funcionalidade	Processo de negócios
Visão	Desenvolvimento de aplicações isoladas	Desenvolvimento de soluções integradas
Estados	Maior número de estados	Menor número de estados
Construção	Dependente da Linguagem	Independente da Linguagem

Fonte: Elaborado pelo próprio autor a partir de dados de OASIS (2007)

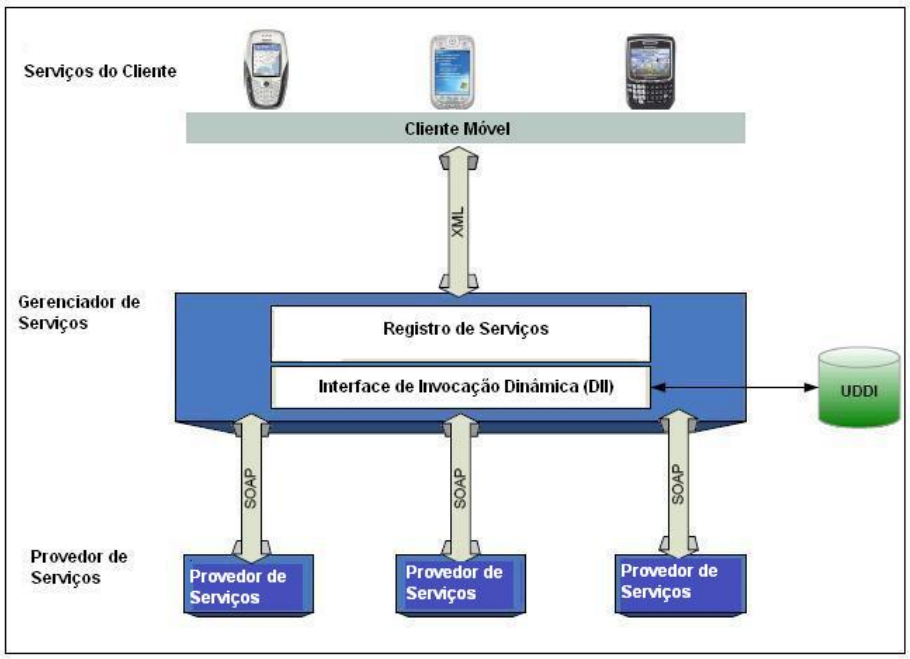
2.3.5 Tecnologias SOA para suporte em Dispositivos Móveis

A arquitetura orientada a serviços para suporte a dispositivos móveis mostrada a seguir foi proposta em (NIELSEN, 2006). Essa arquitetura atende aos seguintes requisitos:

1. Integração dinâmica de novos serviços por provedores;
2. Descoberta dinâmica de serviços disponíveis;
3. O uso de um *software* de fonte aberta para desenvolver a solução.

Esta arquitetura se divide em quatro componentes principais: provedores de serviço, gerenciador de serviços, clientes e o registro UDDI (*Universal Description*

Discovery Integration). A Figura 2.3 apresenta o modelo da arquitetura proposta por (NIELSEN, 2006):



Fonte: Adaptado de NIELSEN (2006)

Figura 2.3: Exemplo de Arquitetura SOA para Dispositivos Móveis usando *Web Services*

Os principais componentes da arquitetura são descritos a seguir:

- **Provedores de Serviço**: oferecem os serviços disponíveis e são descritos utilizando a liguagem WSDL, que é uma linguagem de definição de serviço Web. Ela estabelece o meio pelo qual os provedores de serviços definem a interface para esses serviços. Além disso, a WSDL possui a localização dos serviços e os detalhes de sua interface descritos em XML, define o que cada serviço faz, como eles se comunicam com outros serviços e onde podem ser encontrados. A localização dos serviços é feita por um identificador de recurso uniforme chamado de *Uniform Resource Identifier* (URI), que pode ser acessado pela Internet.

- **Clientes**: representam os usuários de dispositivos móveis que estão interessados nos serviços disponíveis para ele. Os dispositivos móveis trocam informações

normalmente através de mensagens SMS (*Short Message Service*), um serviço disponível em telefones celulares digitais que permite o envio de mensagens curtas.

- **Gerenciador de Serviço**: é responsável pelo fluxo das informações entre os dispositivos móveis e o provedor de serviços. Um gerenciador de serviços é um *web service* que invoca os serviços dinâmicamente. A comunicação entre o gerenciador de serviços e os provedores de serviços é feita usando o protocolo de transporte *Service Oriented Application Protocol* (SOAP) e utiliza a linguagem XML para comunicação com os dispositivos móveis.

- **Registro UDDI(*Universal Description Discovery Integration*)**: ele é utilizado para a localização de novos serviços, facilita a busca ou consulta das interfaces de serviços. O serviço é executado pelo gerenciador de serviços, quando o usuário envia uma requisição de um novo serviço.

2.3.6 Limitações encontradas na SOA

Algumas limitações ou dificuldades encontradas para o uso da SOA são (OASIS, 2007):

- Falta de incentivos desestimula o reuso de serviços;
- Políticas de arquitetura limitada e limitações sobre informações de negócio, valor e custo;
- Falta de interação entre equipes o que limita o reuso;
- Infraestrutura construída sem planejamento.

2.4 *Unified Modeling Language* (UML)

2.4.1 Introdução

No início da última década, foram propostas diversas metodologias para modelagem e projeto de sistemas orientados a objetos. A criação de diversas ferramentas e notações dificultou a disseminação e consolidação das novas propostas. Em 1996, apresentou-se a UML - Linguagem de Modelagem Unificada, criada de forma conjunta pelos principais autores da nova abordagem (JACOBSON, 2004).

Essa padronização contribuiu de forma efetiva para a consolidação dos novos métodos e constituiu em uma nova fase para a Engenharia de *Software*. O maior ganho desta metodologia é o grau de reusabilidade dos módulos especificados. Além disto, permite a inserção de novos requisitos com menor impacto sobre o sistema. Apesar de a UML ser uma padronização para as notações resultantes da modelagem, as técnicas e métodos para fazê-la depende das características do sistema e da equipe de desenvolvimento. O fato de não existir uma notação padronizada para modelar qualquer tipo de aplicação é que deu início ao surgimento da UML (JACOBSON, 2004).

A modelagem é importante não só para sistemas grandes e complexos, mas também para sistemas pequenos e de menor complexidade. Isto porque eles tendem a crescer e a se tornarem mais complexos. Para sistemas grandes e complexos a modelagem é essencial, pois não é possível ter uma visão única de todo o sistema. Dividindo-se o sistema em partes menores e construindo vários modelos, de forma que cada modelo expresse diferentes níveis de abstração, ele será melhor compreendido e representado. A escolha dos modelos também é importante. Cada sistema tem características que tornam alguns modelos mais importantes que outros. Deve-se construir diversos modelos para um sistema e não apenas um.

Muitas empresas partem direto para a programação, na construção de um sistema, devido à falta de tempo para a modelagem ou pela estipulação de um prazo muito curto para a entrega do sistema (BOOCH, 2005). O autor afirma que esta prática leva a sérios problemas na implementação e reduz a qualidade do sistema.

Com erros na implementação do sistema, as empresas tentam contornar o problema aumentando as horas de trabalho dos desenvolvedores. Entretanto, dependendo do tamanho do sistema, estes esforços não solucionarão os erros a tempo. Por isto a modelagem é importante. Ela reduz a probabilidade de erros na implementação, pois documenta a especificação dos requisitos e das funções do sistema.

A UML é um padrão OMG (*Object Management Group*), reconhecido e utilizado pela indústria de *software* entre outras áreas. Esta padronização de modelagem orientada a objetos é devido ao fato da necessidade de modelagem de qualquer sistema, não importando o tipo, seja de forma correta, consistente, simples e de fácil comunicação com outros sistemas. Além disso, é a maneira mais fácil de modelar sistemas, pois ela permite modelar sistemas com seu comportamento, sua arquitetura , seus processos de negócio e estruturas de dados (BOOCH, 2005).

A UML é uma linguagem de modelagem visual de artefatos de sistemas de informação, que representa inúmeras faces de um sistema, que podem ter diferentes níveis de abstração, como nível dos requisitos do cliente, do analista ou do desenvolvedor. Esta visão é obtida através dos diversos diagramas existentes na UML, sendo os principais: diagramas de caso de uso, diagrama de classes, diagrama de seqüência, diagrama de estado e diagrama de componentes (BOOCH, 2005).

Atualmente a UML está na versão 2.0, permitindo uma maior evolução na parte visual, devido ao fato de ter sido acrescentados novos elementos do modelo. Além disso, a nova versão permite novos elementos existentes em tecnologias como MDA e SOA (MCCOY, 2007).

A especificação da UML 2.0 foi dividida em quatro partes:

- Superestrutura: onde é especificado os elementos e os diagramas da UML. Possui o conceito de perfis, estereótipos e valores atribuídos (*tagged values*), que serão apresentados na sessão 2.4.5.

- Infra-estrutura: onde é especificado as classes que formam a superestrutura da UML e da MOF.

- OCL (*Object Constraint Language*): É uma linguagem utilizada para descrever

restrições ou condições aos objetos do modelo.

- *Diagram Interchange*: onde é especificado os diagramas UML entre as ferramentas de modelagem fabricadas pelas empresas. Ela estende o metamodelo UML com um pacote (*package*) que contém informações sobre os gráficos utilizados nos diagramas UML, que devem ser semelhantes aos diagramas criados na UML.

2.4.2 Mecanismos de Extensão

A UML contém alguns conceitos e notações de forma a satisfazer os requisitos de modelagem de sistemas. Porém, podem surgir situações onde são necessários a introdução de outros conceitos ou notações que não foram definidos no documento original da UML. Dessa maneira, a UML possui mecanismos de extensões de forma a estender a UML, estes mecanismos são: restrições, valores atribuídos (*tagged values*) e estereótipos.

Estes mecanismos permitem SILVA (2005):

- A inserção de novos elementos de modelagem que permitem facilitar a compreensão dos modelos UML que serão criados;
- Definir itens padrões que não são considerados suficientemente interessantes ou complexos para serem definidos diretamente como elementos do metamodelo UML;
- Definir extensões específicas para uma determinada tecnologia ou linguagens de programação;
- Alterar a estrutura e semântica dos elementos do modelo.

A UML permite sua ampliação de forma controlada para que possa representar diferenças em modelos em qualquer domínio. Os conceitos dos mecanismos de extensibilidade da UML serão abordados na sessão 2.4.5.

2.4.3 MDA - *Model Driven Arquitecture*

A OMG criou a MDA com o propósito de desenvolvimento independente de plataforma. A MDA separa o que o sistema necessita fazer de como ele pode fazer. Dessa maneira, é possível o desenvolvimento de projetos independente de plataforma, ou seja, a MDA provê modelos independentes de plataforma para diversos domínios de aplicações. Com isso, foi criado o CORBA, que é uma tecnologia que permite conectar qualquer tipo de aplicação com qualquer linguagem orientada à objetos ou plataforma. (FOWLER, 2003).

Com o passar dos anos, a OMG concluiu que as tecnologias existentes no mercado eram bastante diversificadas, e a medida que o tempo passava as empresas eram forçadas a suportar novas linguagens de programação, outras plataformas ou sistemas operacionais, além de uma grande variedade de protocolos de rede, para permitir a interoperabilidade. Para conseguir tal objetivo, a OMG deveria criar um sistema padronizado, onde os modelos sejam independentes de linguagem, sistema ou protocolo, onde surgiu a MDA (*Model Driven Architecture*) (FOWLER, 2003).

A MDA propõe um modelo independente de plataforma e de sistema operacional. Dessa forma, se for introduzido um novo sistema operacional ou uma nova linguagem de programação no sistema, deve-se apenas reutilizar o modelo independente de plataforma (PIM-*Plataform Independent Model*) que vai gerar um novo modelo, específico para a nova aplicação. Assim, se forem introduzidas novas tecnologias, as aplicações existentes são mantidas, documentadas como modelos independentes de plataforma e podem ser reutilizadas.

2.4.4 MOF - *Meta Object Facility*

A OMG define duas possibilidades para definir linguagens de um domínio específico, sendo o primeiro baseado na definição de uma nova linguagem, alternativa à UML. Para definir uma nova linguagem são usados os mesmos mecanismos utilizados pela OMG para a definição da UML e seus metamodelos. Dessa forma, é preciso descrever seu metamodelo utilizando a MOF*(Meta Object Facility)*, uma linguagem feita especialmente para definir linguagens de modelagem orientada a objetos e também permite a construção de ferramentas que auxiliam nesta atividade.

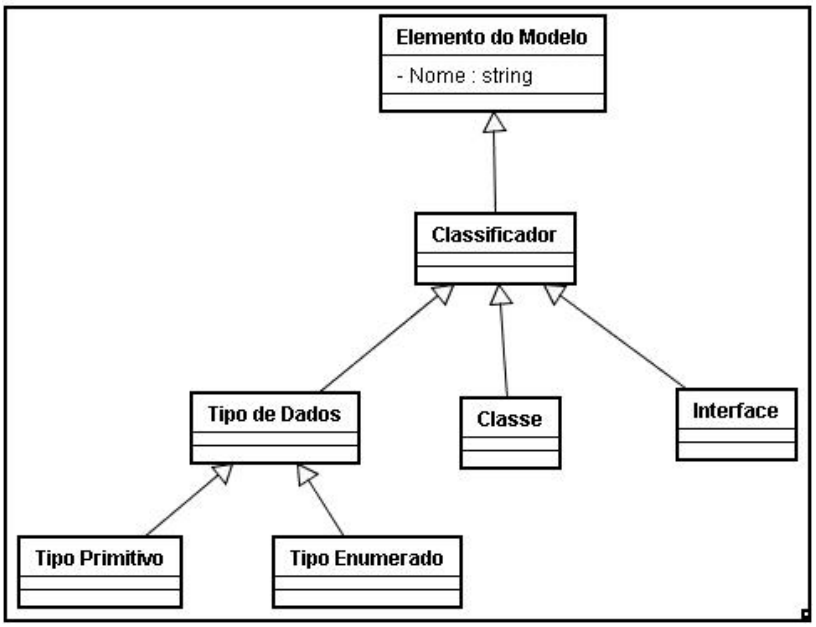
Por outro lado, a segunda alternativa é estender a UML utilizando um conjunto de mecanismos de extensões (estereótipos, valores atribuídos e restrições), para especializar seus elementos para uma aplicação de domínio específico (FOWLER, 2003).

O papel de um metamodelo é definir a semântica de como os elementos do modelo podem ser instanciados, assim como em uma linguagem de modelagem, são definidos a mesma estrutura, a semântica e as restrições para um grupo de modelos. Como um metamodelo é também um modelo, ele precisa ser especificado por uma linguagem de modelagem, ou um metamodelo.

O modelo que contém os elementos que permitem a criação de metamodelos é denominado metametamodelo, que é responsável pela definição de uma linguagem para a especificação de metamodelos. A OMG criou a MOF, uma linguagem para a especificação de metamodelos. Esta linguagem contém elementos básicos para a criação de metamodelos. Os metamodelos da UML é baseado no meta-metamodelo MOF (FOWLER, 2003).

O metamodelo UML descreve como um modelo UML é estruturado. A Figura 2.4, demonstra um metamodelo UML simplificado, onde temos a classe Elemento do Modelo, que é uma superclasse comum da UML. Um Elemento do Modelo não tem uma notação definida, possui um nome e as subclasses de Elemento do Modelo possui a descrição dos nomes dos elementos com notação definida. Os construtores abstratos são aqueles que não podem ser usados diretamente pelo utilizador e servem para dar estrutura e organização ao metamodelo UML, tais como: Elemento do Modelo, Classificador. Por outro lado, os construtores concretos são aqueles que são usados

ou instanciados diretamente pelo utilizador, tais como classes, atributos, operações e associação (ERIKSON, 2004).



Fonte: Adaptado de ERIKSON (2004)

Figura 2.4: Metamodelo simplificado da UML

A metaclasse abstrata Classificador, é um supertipo definido no metamodelo UML que é usado para definir a classificação dos elementos do metamodelo UML, descreve estrutura e comportamento, ou seja, sempre que se pretende referir indistintamente a qualquer dos seguintes conceitos: classe, interface, tipo de dados, caso de utilização, ator, sinal, subsistema, nó ou componente. Ela é uma generalização de uma Classe, Interface e Tipo de Dados. O Classificador Classe possui atributos e operações e o Classificador abstrato Tipo de Dados, define as estruturas de dados básicas da linguagem, tais como Tipos de Dados Primitivos (*integer, string, boolean*) e Enumerados, que permite o usuário definir um conjunto de valores ou elementos do modelo. O Classificador declara um conjunto de características e obrigações. Uma Interface é produzida de acordo com um Classificador (ERIKSON, 2004).

Os modelos UML devem cumprir estas regras para ser uma instância correta da UML, por isso pode-se utilizar modelos UML conhecendo exatamente sua estrutura e representação. A MOF é a principal tecnologia para MDA, oferece os conceitos

e ferramentas em relação aos modelos padronizados. Usando a MOF sobre uma linguagem de modelagem qualquer pode-se definir as transformações entre os modelos, escrito em qualquer linguagem. Sem uma linguagem padrão para descrevermos metamodelos (como por exemplo a UML), as tranformações não poderiam ser utilizadas corretamente e dificultaria a utilização da MDA (SILVA, 2005).

2.4.5 Perfil UML

A OMG criou um mecanismo para possibilitar a modelagem de artefatos de mais alto nível permitindo utilizar a UML para atender necessidades específicas de acordo com as tecnologias. Esse mecanismo é chamado de perfil, que permite estender a UML através de estereótipos, valores atribuídos (*tagged values*) e restrições que permitem especializar a UML para modelar aplicações de domínio específico (RUMBAUGH, 2004).

Entretanto, existem situações em que a UML não é apropriada para modelar aplicações de um domínio específico. Isso acontece quando a sintaxe e a semântica da UML não permitem expressar os conceitos específicos do domínio, ou quando se deseja restringir e especializar os elementos da UML que são genéricos e numerosos.

Com os Perfis UML, é possível criar elementos que expressam conceitos específicos de um domínio específico, sem que haja a necessidade de criação de uma nova linguagem de modelagem (FERNANDÉZ, 2004). Além disso, é possível continuar a utilizar os elementos básicos e abstratos da linguagem, porém, sem ficar restrito somente a estes elementos. Por exemplo, é preferível modelar um sistema Java utilizando componentes JEE(*Java Plataform Enterprise Edition*), que normalmente utilizam EJB(*Enterprise JavaBeans*), do que modelar componentes genéricos utilizando a classe *Component* do metamodelo UML.

A utilização do perfil UML permite a criação de novas linguagens de modelagem utilizando linguagens de modelagem baseadas no meta-metamodelo MOF. Um fator importante é que esta nova linguagem preserva a semântica dos elementos do metamodelo existente e adiciona novas regras ao metamodelo UML (FOWLER, 2003). A extensão, portanto não afeta o metamodelo e os modelos existentes antes da criação

do perfil UML. Outro fator importante é que a nova linguagem criada através de um perfil UML pode ser utilizada em ferramentas que dêem suporte a este mecanismo, sem que haja a necessidade de adaptação destas ferramentas. Dessa forma, facilita automatizar o processo da transformação do modelo em código-fonte, onde um componente EJB poderá ser convertido em código-fonte para a criação deste componente em cima desta tecnologia.

Segundo Rumbaugh (2004), um perfil é um pacote que identifica um subconjunto de um metamodelo base existente. Para a criação de um perfil, são utilizados estereótipos, valores atribuídos (*tagged values*) e restrições.

- **Estereótipos:** Um estereótipo é uma extensão para o vocabulário de uma linguagem que classifica os elementos do modelo de acordo com o comportamento, restrições e valores atribuídos (*tagged values*) (RUMBAUGH, 2004). Ele é um elemento de modelagem baseado em elementos de modelagem já existentes. A informação contida em um estereótipo é a mesma contida nos elementos base do modelo, permitindo que ferramentas armazenem e manipulem um novo elemento do mesmo modo que faz com os elementos já existentes. Um estereótipo é definido por um nome e por um conjunto de elementos de um metamodelo. Ele tem a indicação da sua classe base, normalmente com o nome do estereótipo entre « » e quando é possível, uma representação gráfica (ícone) correspondente. A classe base de um estereótipo é uma classe no metamodelo UML tal como *Class, Association ou Refinement.* Na UML 2.0, os estereótipos podem definir tudo o que uma classe define, incluindo atributos e associações. Várias ferramentas CASE para UML oferecem a possibilidade de escolher entre uma ou mais formas de visualizar um estereótipo.

- **Valores atribuídos(*tagged values*):** São considerados como uma extensão da propriedade de um elemento do modelo. Uma Classe possui nome, visibilidade, persistência e outros atributos associados a ela. Desse modo, valores atribuídos é a definição de uma nova propriedade que pode ser associada com um elemento do modelo. Por exemplo, cada *tagged value*, ou propriedade, possui um nome, um

tipo e um valor inicial se for necessário. Por exemplo, Data:*date*= 01/01/2007, esta expressão se refere ao nome "Data", com o tipo "*date*" e o valor inicial igual a "01/01/2007".

- **Restrições:** As restrições são extensões para a semântica da linguagem. Uma restrição deriva diretamente de um elemento de um modelo, é uma forma de expressar uma regra específica para um determinado elemento do perfil. É previsto a utilização de restrições no modelo descrevendo as restrições semânticas que esse modelo tem de satisfazer. Especifica as condições que devem ser atendidas, permitindo acrescentar novas regras para o modelo ou modificar as já existentes. As restrições podem ser expressas em qualquer linguagem programação, como por exemplo: tempo <= 2 e também em linguagem natural, como por exemplo: o tempo deve ser menor ou igual a 2 segundos. Restrições podem ser escritas em linguagem formal de forma padronizada chamada *Object Constraint Language*(OCL), que será aresentada na próxima sessão (OCL, 2003).

2.4.6 OCL (*Object Constraint Language*)

A OCL é uma linguagem formal de expressões para especificar restrições sobre modelos orientados a objetos ou outros artefatos da linguagem UML. Dessa forma, ela pode estar disponível em ferramentas CASE UML.

Alguns benefícios da OCL são que ela é formada de elementos do modelo como atributos, operações e associações. Além disso, ela representa constantes e operadores. Ela expressa o que a restrição representa no sistema e não como ela é implementada (OCL, 2003).

As expressões OCL são utilizadas para definir condições invariantes nas classes representadas em um modelo e também são utilizadas para especificar as pré e pós-condições em operações aplicadas a classes deste modelo (OCL, 2003).

Cada expressão OCL fornece as restrições ligadas a um elemento do modelo. Para isso, deve-se informar qual é o contexto ou elemento que a expressão OCL se refere. O contexto de uma expressão pode ser uma classe de objetos ou uma operação

de um objeto. Para representar um contexto em OCL utilizamos a palavra reservada *context*(contexto). Numa expressão em OCL, a palavra reservada *self* é utilizada para referenciar uma instância do contexto (OCL, 2003).

Invariantes são restrições condicionais que se aplicam a todas as classes e deve resultar sempre como sentença verdadeira para não criar inconsistência no modelo. Os objetos do modelo devem respeitar essas condições durante toda sua existência no sistema. Em OCL, para indicar que uma expressão é uma invariante, utilizamos a palavra reservada *inv*, após a declaração do contexto (OCL, 2003). Por exemplo, toda pessoa deve ter pelo menos dezoito anos de idade.

context pessoa **inv**:
self.idade > 18

Pré-condições são restrições que mostram o estado no qual o sistema deve estar. Podem ser definidas em métodos do modelo, sendo utilizada antes da execução do método associado ao objeto. É utilizado para a validação dos parâmetros de entrada de um método. Em OCL, quando queremos expressar uma condição na forma de pré-condição, utilizamos a palavra reservada *pre* após a declaração do contexto. Por exemplo, o placar de pontos é dado ao jogador somente se o mesmo jogar até o final do jogo. Pós-condições são restrições que apresentam o estado do sistema após a

context jogador::caculapontuação()::integer **pre**:
self.fimdojogo = true
resultado = 5000

execução de um método de um objeto. É utilizada para a validação dos parâmetros de saída de um método, ou para descrever como os valores de entrada são modificados após a execução de um método. Em OCL, declaramos uma expressão na forma de pós-condição utilizando a palavra reservada *post* após a declaração do contexto. Por exemplo, o salário de uma pessoa aumenta de acordo com uma data.

context pessoa::comecou(d:date)::integer **post**:
if d > "01/01/2007" **then**
    resultado = 5000
else
   resultado = 4000

Capítulo 3

O Perfil *Mobile Services*

3.1 Introdução

O Perfil *Mobile Services* foi criado com o objetivo de definir uma extensão constituído por um grupo de estereótipos, valores atribuídos *(tagged values)* e restrições, que especializam a UML para lidar com os conceitos específicos de um domínio de aplicação em particular.

O objetivo deste perfil é definir um conjunto de extensões UML, já que esta não possui uma maneira específica para modelar conceitos e práticas no domínio de serviços para ambientes móveis. O perfil proposto é de fácil de utilização sendo possível utilizá-lo em ferramentas CASE que suportem a criação de perfis UML. Dessa maneira, esse perfil permite representar de forma clara os conceitos básicos para o desenvolvimento de serviços para dispositivos móveis através de estereótipos que estendem a UML.

Esta proposta de modelo poderia ser feita utilizando diretamente a linguagem MOF, a mesma linguagem usada para a criação do metamodelo UML. Porém, a MOF exige uma extensa especificação de detalhes. Além disso, necessitaria incluir o modelo às ferramentas CASE UML, para isso é necessária a interação com os fabricantes dessas ferramentas. Também não seria possível utilizar outra metodologia de modelagem sem

utilizar padrões OMG, pois dificultaria a padronização entre as diversas ferramentas de modelagem.

As restrições deste perfil foram especificadas através de texto comum ou OCL quando possível. Não foi possível testar os códigos escritos em linguagem OCL devido ao fato de não possuir uma ferramenta CASE Livre ou gratuita que suporta a utilização de OCL com os metamodelos utilizados neste perfil. A utilização da OCL e sua validação poderão ser realizadas em trabalhos futuros.

3.1.1 Requisitos do perfil proposto

Para que se possa atingir os objetivos deste trabalho, foram definidos alguns requisitos para facilitar a utilização do perfil proposto:

- Um diagrama deverá ser capaz de modelar serviços em ambientes móveis;
- Os diagramas deverão ser de fácil compreensão e utilização, não havendo necessidade de muito trabalho adicional;
- A representação gráfica (ícones) utilizada nos estereótipos dos diagramas, devem representar os tipos de serviços.
- O modelo deverá seguir o metamodelo UML, sendo de fácil compreensão, para que outras pessoas que não são da área de desenvolvimento de sistemas possam compreender o modelo proposto;
- O modelo deverá ser independente de tecnologia ou linguagem de programação;
- O modelo deverá prever a utilização de restrições de cada estereótipo de serviços, estaticamente ou dinamicamente através de expressões textuais ou OCL quando possível.

3.2 Perfil *Mobile Services*

Este perfil define um conjunto de estereótipos, valores atribuídos (*tagged values)* e restrições, que possibilitam uma maneira específica de modelar serviços em ambientes móveis. Eles são aplicados em componentes específicos para este tipo de aplicação, sendo descritos através de diagramas UML.

O perfil é constituído de nome do estereótipo, as classes bases do metamodelo, a descrição, valores atribuídos*(tagged values)* e restrições. Além disso, é apresentada uma representação gráfica dos elementos (ícones), quando possível.

Este perfil possui três estereótipos de classe (*Application Service, Mobile Service, LBS Service.* Os estereótipos foram definidos para possibilitar que no futuro, ele seja transformado em *software* executável através de um *framework* que irá instanciar cada estereótipo com suas respectivas propriedades, métodos e restrições. Além disso, o *framework* deverá fazer a validação e presistência dos conteúdos dos controles ou seja, validar os atributos, métodos e restrições de cada estereótipo.

O perfil apresentado na Figura 3.1, tem por objetivo modelar os serviços utilizados em sistemas móveis com os principais componentes da arquitetura utilizados para esta finalidade. Estes componentes são os provedores de serviços, que implementam e oferecem os serviços disponíveis. Este problema é resolvido pela extensão representada pelos estereótipos *Mobile Service* e *LBS Service*, onde o primeiro é responsável pela modelagem de serviços simples e o segundo é responsável pela modelagem de LBS. Todos os dois tipos de serviços levam em conta as restrições temporais impostas pela mobilidade. Além disso, são especializadas pelo estereótipo de classe *Application Service*, onde são herdados seus atributos e métodos.

Os usuários móveis estão interessados nos serviços disponíveis para ele no ambiente que ele se encontra, sendo que estas informações relativas aos serviços devem ser apresentadas de forma clara e objetiva. Por fim, o Serviço de Aplicação é responsável pelo fluxo de informações entre dispositivos móveis e os serviços, permitindo que eles se comuniquem.

O estereótipo *Application Service*, representa o serviço de aplicação, que é um

Web Service que permite que os serviços se comuniquem com outros provedores de serviços dinamicamente, quando o usuário requisita um novo serviço. Esse estereótipo estende a metaclassse *Class*, que permite instanciar outras classes.

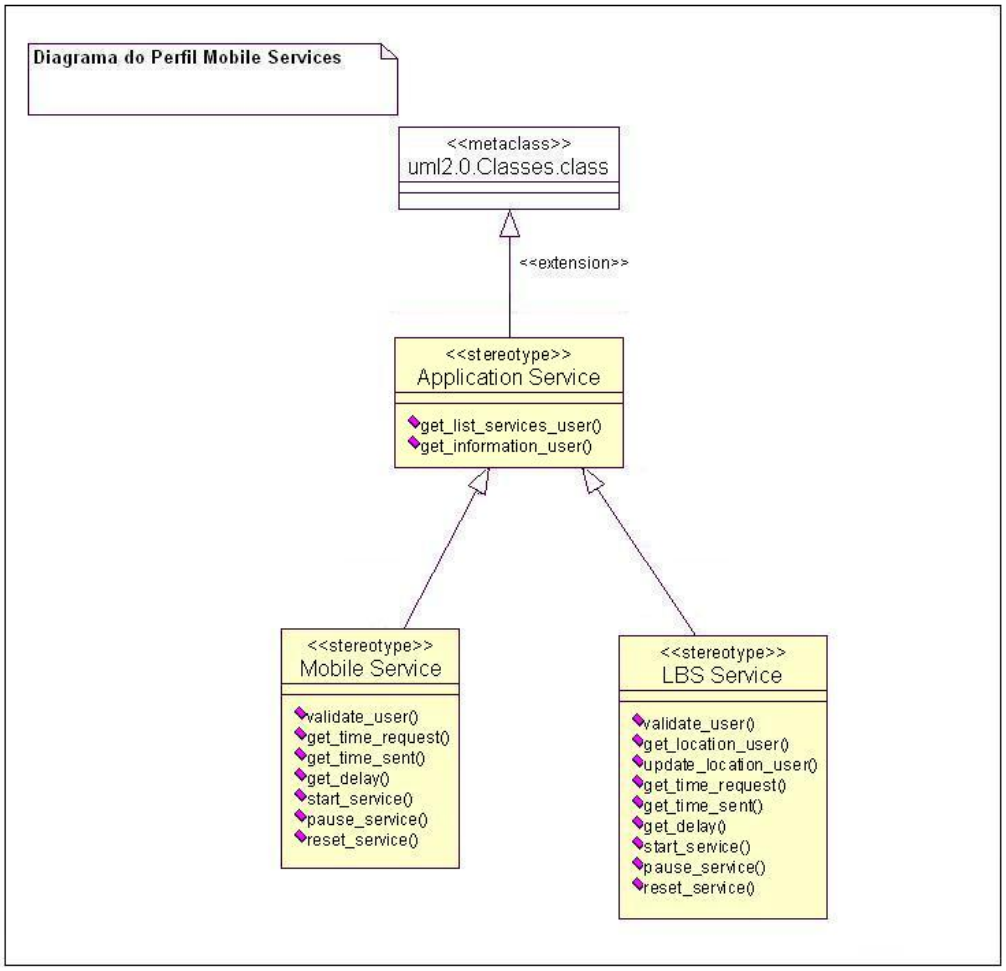


Figura 3.1: Diagrama do perfil *Mobile Services*

O estereótipo *Application Service* é estendido pelos estereótipos *Mobile Service, LBS Service*. Todos os estereótipos criados nesse perfil serão apresentados detalhadamente na próxima sessão.

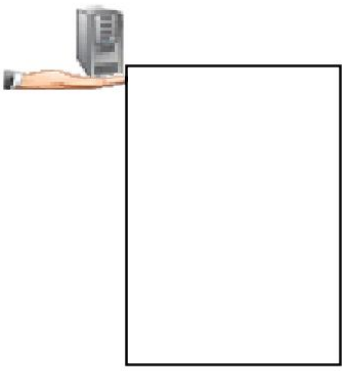
Dessa maneira, conseguimos resolver os seguintes problemas quanto a modelagem de serviços em ambientes móveis, como por exemplo modelar serviços que dependem ou não da localização do usuário móvel e restrições temporais impostas pela mobilidade.

3.2.1 Estereótipos

Os estereótipos representam o principal mecanismo de extensão, onde é possível especializar a UML com elementos já existentes (ERIKSON, 2004). Ele classifica os elementos do modelo de acordo com o comportanmento, restrições e valores atribuídos(*tagged values*) (RUMBAUGH, 2004). A seguir são apresentados os estereótipos criados no perfil *Mobile Services*.

1. Estereótipo: ***Application Service***

   - **Metamodelo:** Classe
   - **Descrição:** Faz a busca dos serviços autorizados para cada usuário, deixando-os disponíveis para serem acessados. Informa para o usuário autorizado o resultado (saída) do serviço logo após o processamento da consulta.
   - **Ícone:**



- **Restrições:** Não aplicável.

- **Valores atribuídos *(tagged values)*:**

  ***get_ list_ services_ user()*:** Método que busca os serviços autorizados para cada usuário.

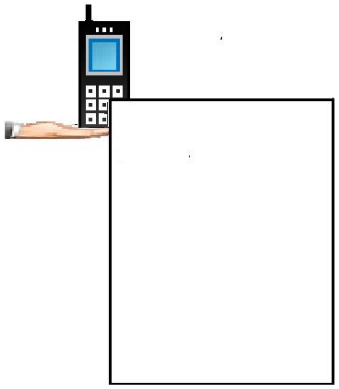
  ***get_ information_ user()*:** Método que busca o conteúdo relativo às informações obtidas do serviço que foi solicitado pelo usuário móvel autorizado.

2. Estereótipo: ***Mobile Service***

- **Metamodelo:** Classe

- **Descrição:** Faz a busca em uma base de dados de assinantes dos serviços e informa se o usuário móvel está ou não autorizado a utilizar o serviço. Caso o usuário móvel tenha permissão de acesso ao serviço, inicia o processamento do serviço que foi solicitado pelo usuário móvel. O serviço é interrompido caso houver algum tipo de interrupção e é reiniciado logo em seguida. Este estereótipo é aplicado à situações independente da localização do usuário móvel.

- **Ícone:**



- **Restrições:**

  - O serviço deve ser interrompido (cancelado) sempre que houver uma condição de *timeout* por algum motivo como falhas na conexão, *handovers*, falta de energia, etc., que podem provocar falha no serviço que está sendo solicitado no momento. Essa condição se dá quando a diferença de tempo entre o instante atual e o momento da requisição do serviço ultrapassar o atraso (*delay*) máximo permitido.

Em OCL temos:

context Mobile Service **inv:**

(time_sent - time_request)<=delay

- **Valores atribuídos *(tagged values)*:**

validate_user(): Método que verifica se o usuário móvel está ou não autorizado a utilizar o serviço que foi solicitado.

get_time_request(): Método que busca a hora da solicitação do serviço.

get_time_sent(): Método que busca a hora atual que a informação do serviço foi enviada para o usuário móvel.

get_ delay(): Método que busca o atraso máximo permitido para um determinado serviço.

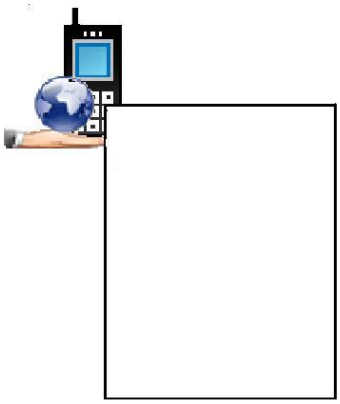
start_ service(): Método que inicia o processamento do serviço.

pause_ service(): Método que cancela o processamento do serviço, devido algum motivo (falhas de conexão, falta de energia, etc.).

reset_ service(): Método que reinicia o processamento do serviço, pelo fato do serviço ser interrompido, descartando a informação a ser enviada ao usuário móvel, começando o processo novamente.

3. Estereótipo: ***LBS Service***

- **Metamodelo:** Classe
- **Descrição:** Faz a busca em uma base de dados de assinantes e informa se o usuário está ou não autorizado a utilizar o serviço. Busca também a localização do usuário, devido ao fato do serviço depender da sua localização. Devido ao fato do usuário poder estar em movimento, é feita uma atualização frequente da sua localização. É interrompido o serviço quando houver algum tipo de de interrupção e é reiniciado logo em seguida. O serviço é cancelado quando a hora do serviço enviado para o usuário ultrapassar o atraso máximo permitido.
- **Ícone:**



- **Restrições:**

  - O serviço deve ser interrompido (cancelado) sempre que houver uma condição de *timeout* por algum motivo como falhas na conexão, *handovers*, falta de energia, etc., que podem provocar falha no serviço que está sendo solicitado no momento. Essa condição se dá quando a diferença de tempo entre o instante atual e o momento da requisição do serviço ultrapassar o atraso (*delay*) máximo permitido.

  Em OCL temos:

  context Mobile Service **inv:**
  (time_sent - time_request)<=delay

- **Valores atribuídos *(tagged values)*:**

  validate_user(): Método que verifica se o usuário móvel está ou não autorizado a utilizar o serviço que foi solicitado.

  get_location_user(): Método que verifica a localização do usuário móvel.

  update_location_user(): Método que faz a atualização da localização do usuário a medida que ele se desloca.

  get_time_request(): Método que busca a hora da solicitação do serviço.

  get_time_sent(): Método que busca a hora atual que a informação do serviço foi enviada para o usuário móvel.

  get_delay(): Método que busca o atraso máximo permitido para um determinado serviço.

  start_service(): Método que inicia o processamento do serviço.

  pause_service(): Método que cancela o processamento do serviço, devido algum motivo (falhas de conexão, falta de energia, etc.).

reset_ service(): Método que reinicia o processamento do serviço, pelo fato do serviço ser interrompido, descartando a informação a ser enviada ao usuário móvel, começando o processo novamente.

3.2.2 Resumo do Perfil *Mobile Services*

A tabela 3.1 descreve o resumo do perfil proposto com o metamodelo, *tagged values*, restrições e uma breve descrição dos estereótipos criados no perfil.

Tabela 3.1: Resumo do Perfil *Mobile Services*

Estereótipo	**Metamodelo**	**Restrições**	**Descrição**
«Application Service»	Classe	-Não aplicável.	- Faz a busca dos serviços autorizados para cada usuário móvel, informando o resultado do serviço.
«Mobile Service»	Classe	(time_sent - time_request)<=delay	- É utilizado para serviços independente da localização do usuário móvel. - O serviço é interrompido (cancelado) quando a diferença entre a hora que o serviço foi solicitado e a hora atual ultrapassar o atraso máximo permitido.

Estereótipo	**Metamodelo**	**Restrições**	**Descrição**
«LBS Service»	Classe	(time_sent - time_request)<=delay	- É utilizado para serviços baseados na localização do usuário móvel. - O serviço é interrompido (cancelado) quando a diferença entre a hora que o serviço foi solicitado e a hora atual ultrapassar o atraso máximo permitido.

Capítulo 4

Utilização do Perfil *Mobile Services*

4.1 Introdução

Modelos permitem a omissão de detalhes menos importantes e o realce de características fundamentais do sistema. Um bom modelo é uma alternativa eficiente para guiar o projeto do sistema, minimizando o custo e os riscos do mesmo. Os modelos são mapeados para facilitar a implementação e especificação do domínio do problema independente da tecnologia a ser utilizada.

Um negócio pode ser definido como um conjunto de atividades, visando um objetivo específico. Um modelo de negócio é a representação real do sistema para todos os membros envolvidos, pessoas ou sistemas (BMP, 2007). O objetivo do modelo é permitir a utilização em processos de desenvolvimento de *software* e em ferramentas que permitem a utilização desse recurso (BMP, 2007).

Na modelagem baseada em SOA, os principais requisitos viram serviços e estes serviços são acessados por outros serviços, modularizando e aumentando o número de interfaces dos serviços, limitando o desempenho. A UML permite projetar a arquitetura orientada a serviços.

Neste capítulo foi feita a utilização do perfil proposto em um sistema real através de um estudo de caso. Para isso, instanciou-se o perfil UML criado. Por

fim foi feita a modelagem de um caso de uso de serviços comuns e outro de serviços baseados na localização do usuário móvel.

4.2 Estudo de Caso: Sistema PVAnet

4.2.1 Breve descrição do Sistema

O PVANet é um sistema de gerenciamento de conteúdo acadêmico da Universidade Federal de Viçosa (UFV). O sistema é utilizado por alunos e professores da UFV.

A Figura 4.1, mostra a interface de login do sistema para *desktop*, pois ainda não possui interface para dispositivos móveis. Dentre as funcionalidades principais do sistema estãos: A lista de disciplinas matriculadas pelo aluno; o conteúdo das disciplinas matriculadas com material disponível para *download*. Possui também tarefas de cada disciplina matriculada; notícias de uma determinada disciplina que o aluno está matriculado.

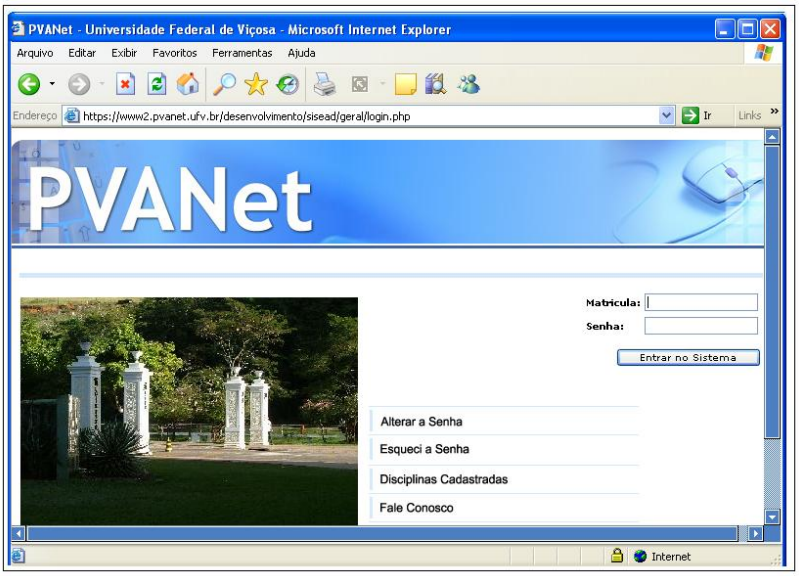


Figura 4.1: Tela de Login do Sistema PVANet

O sistema PVANet não possui nenhuma funcionalidade para serviços em ambientes móveis ou baseados na localização do usuário. Por esse motivo, foram acrescentados dois casos de usos: "Exibir informação departamento" e "Localizar evento disciplina", para que se possa fazer a modelagem desses tipos de serviços propostos neste perfil. Os casos de usos acrescentados foram descritos nas próximas sessões.

4.3 Caso de uso: Exibir informação departamento

O aluno deseja acessar o serviço "Exibir informação departamento" e verificar informações relativas a qualquer departamento da Universidade, independente da sua localização geográfica. Ao acessar o serviço o aluno tem acesso a um mapa da Universidade com todos seus departamentos, onde ele pode clicar no mapa e saber informações sobre os mesmos. O serviço fornece informações ao aluno sobre o departamento, como por exemplo o nome, breve descrição, disciplinas lecionadas e seus laboratórios.

Breve descrição: O aluno usa o serviço para exibir informações sobre algum departamento da Universidade.

Ator: Aluno.

Pré-condições: O aluno deve estar autorizado para utilizar o serviço.

Fluxo Principal:

1. O Aluno clica sobre um departamento no mapa da Universidade.

2. O serviço exibe o nome do departamento, breve descrição, disciplinas do departamento e seus laboratórios.

3. O aluno visualiza as informações sobre o departamento escolhido.

Fluxo Alternativo:

a. O aluno pode visualizar as informações sobre os departamentos como visitante.

Fluxo de Excessão:

a. O serviço deve ser interrompido (cancelado) sempre que houver uma condição de timeout por algum motivo como falhas na conexão, handovers, falta de

energia etc., que podem provocar uma falha no serviço que está sendo solicitado no momento. Essa condição se dá quando a diferença de tempo entre o instante atual e o momento da requisição do serviço ultrapassar o atraso (delay) máximo permitido.

Pós-Condições: O Aluno obteve as informações disponíveis sobre os departamentos que deseja visualizar.

A utilização do perfil proposto será feito através do diagrama de classes e sequência apresentados nas próximas sessões deste capítulo.

4.3.1 Diagrama de Classes

Nos diagramas de classes a seguir, alguns atributos (campos) relevantes como: (nome, CPF, etc.), não foram acrescentados pelo fato de não serem necessários para a compreensão do problema.

Na Figura 4.2, os serviços são acessados pela classe aluno através de uma interface, representada pela classe *Form*. A classe *Form* utilizada, foi do perfil UML para modelagem de aplicações *Web*, proposto por (CONALLEN, 1999).

O serviço de aplicação, representado pela classe *Application Service* possui os métodos *get_list_services_user()* para buscar os serviços que cada usuário móvel está autorizado acessar, e o método *get_information_user* para buscar as informações obtidas em cada serviço. Os serviços são acessados através da classe *Form* se comunica com o serviço "Exibir informação departamento", através da classe *Information_department*, que por sua vez, herda os atributos e métodos da classe *Mobile Service*.

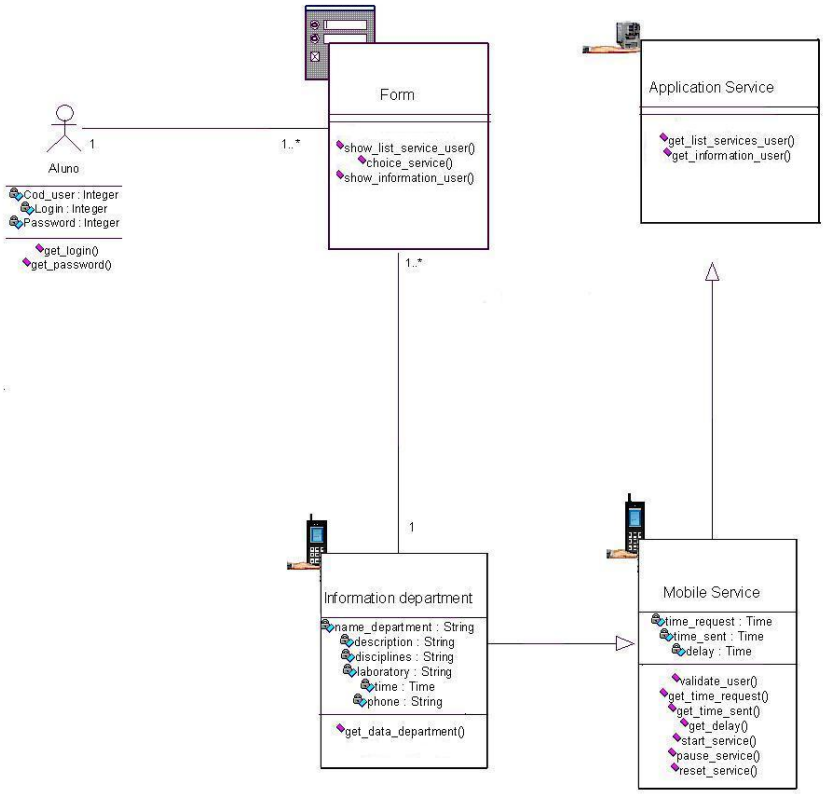


Figura 4.2: Diagrama de Classes do Serviço "Exibir informação departamento"

A classe *Mobile Service* tem em especial os atributos do tipo *time*, que guardam a hora que o serviço foi solicitado, a hora que ele foi atendido e o atraso (*delay*) máximo permitido para o serviço ser processado, evitando que a informação não tenha mais valor devido ao fato do usuário móvel poder estar em movimento, ou que aconteça alguma interrupção no sistema.

Além disso, a classe *Mobile Service* é responsável por atender os serviços comuns ao usuário móvel independente da sua localização. Ela busca na classe *Information department* o serviço fornece informações sobre algum departamento da Universidade.

A classe *Information department* possui um método em especial: *get_ data_ department()*, método que busca os dados do departamento para informar ao usuário móvel. Além

disso, possui os dados dos usuários que podem acessar o serviço, que são validados pelo método *validate_ user()* da classe *Mobile Service*.

Podemos ver que existe uma associação somente entre as classes Aluno, *Form* e *Information department*, pois esta última é uma extensão da classe *Mobile Service*, sendo também um serviço móvel específico, que por sua vez, é uma extensão da classe *Application Service*, portanto possui as mesmas características e funções destas classes, onde são herdados todos os seus atributos e métodos.

Dessa maneira, para inserir um novo serviço no modelo, basta inserir uma classe do novo serviço desejado somente com suas características e funções específicas, ou seja, com seus próprios atributos e métodos, já que as outras classes propostas no perfil já estão prontas, bastando repetir o uso delas.

4.3.2 Diagrama de Sequência

O diagrama de sequência apresentado na Figura 4.3 é um diagrama de interação entre os objetos, mostrando uma visão temporal das rotinas do caso de uso "Exibir informação departamento". Os objetos participantes da interação são organizados na horizontal. Abaixo de cada objeto existe uma linha (linha de vida) que indica quando o objeto está fazendo algo.

Os objetos trocam mensagens (operações entre objetos) e são representadas com linhas horizontais partindo da linha de vida do objeto remetente e chegando a linha de vida do objeto receptor, cada linha horizontal representa um evento. A direção das setas nas mensagens permite deduzir a ordem na qual elas são enviadas, por esse motivo este tipo de diagrama é chamado de diagrama de sequência.

Este diagrama foi feito somente entre as classes Aluno, *Form* e *Information department*, que representa a associação no diagrama de classes apresentado anteriormente. As outras classes não foram utilizadas pelo fato de ter sido utilizado as extensões propostas neste perfil, como foi explicado anteriormente.

Além disso, é possível representar as restrições de cada evento através de texto comum e linguagem OCL. A linguagem OCL é representada entre chaves.

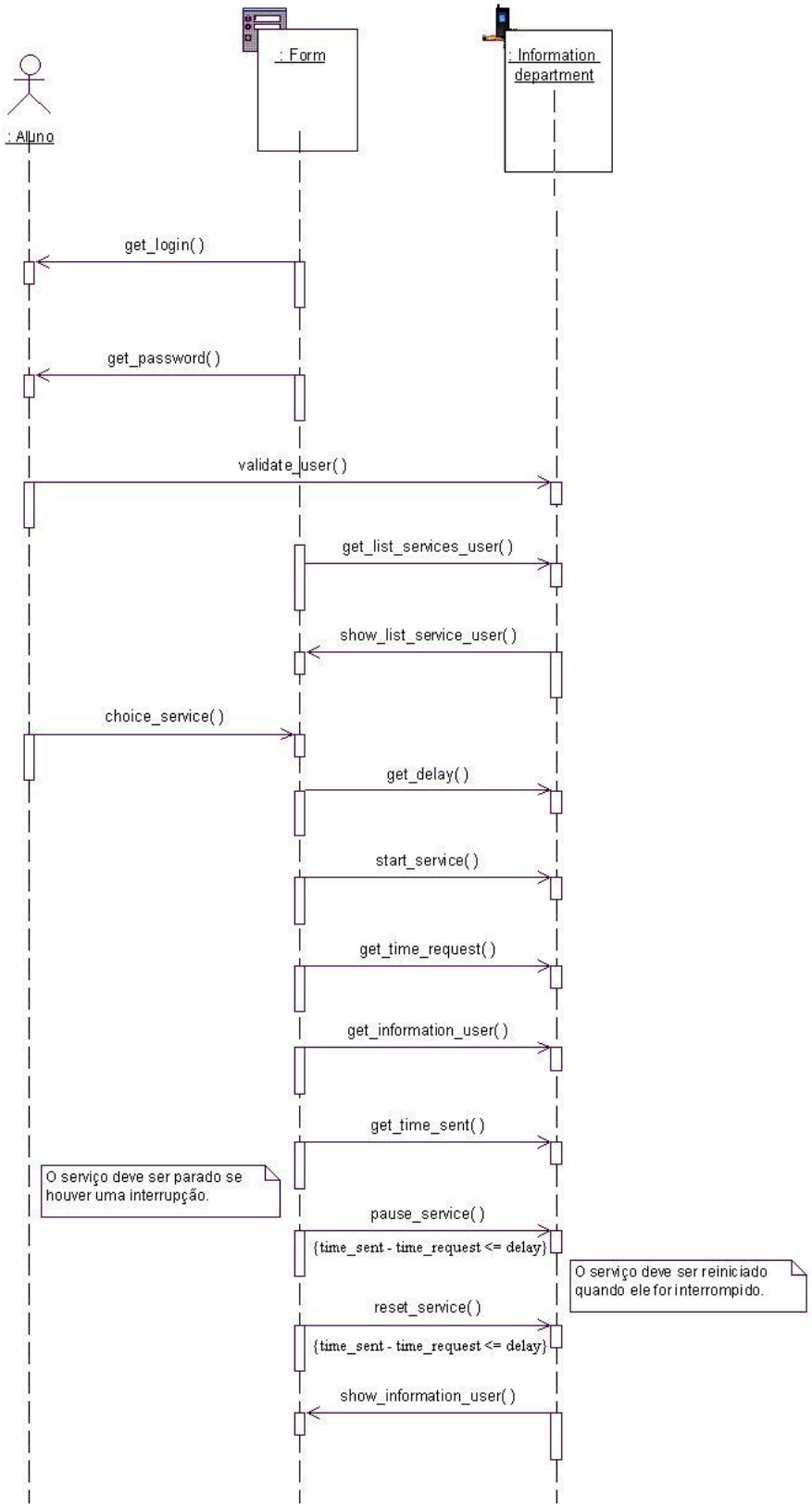


Figura 4.3: Diagrama de Sequência do Serviço "Exibir informação departamento"

4.4 Caso de uso: Localizar evento da disciplina

O aluno deseja acessar o serviço "Localizar evento disciplina" que informa ao aluno a localização de um evento escolhido a partir da sua localização geográfica.

Breve descrição: O aluno usa o serviço para localizar onde será o evento da disciplina, como por exemplo aula expositiva, seminários, congressos relacionados com a disciplina, laboratórios, etc.

Ator: Aluno.

Pré-condições: O aluno deve estar autorizado para utilizar o serviço.

Fluxo Principal:

1. O Aluno solicita a localização do evento da disciplina que está matriculado em relação a sua localização atual.

2. O serviço exibe o código, o nome da disciplina e a localização do evento da disciplina através de um mapa.

3. O aluno visualiza a localização do evento da disciplina em relação a sua localização geográfica.

Fluxo Alternativo:

a. O aluno pode visualizar a localização dos eventos das disciplinas que foram cursadas ou a serem cursadas como visitante.

b. A localização do aluno é frequentemente atualizada pelo serviço.

Fluxo de Excessão:

a. Se o aluno não estiver matriculado, a localização não será informada ao aluno.

b. O serviço deve ser interrompido (cancelado) sempre que houver uma condição de *timeout* por algum motivo como falhas na conexão,*handovers*, falta de energia etc., que podem provocar uma falha no serviço que está sendo solicitado no momento. Essa condição se dá quando a diferença de tempo entre o instante atual e o momento da requisição do serviço ultrapassar o atraso (*delay*) máximo permitido.

Pós-Condições: O Aluno obteve a localização dos eventos relacionados com

as disciplinas que deseja visualizar.

4.4.1 Diagrama de Classes

No diagrama de classes apresentado na Figura 4.4, os serviços são acessados pela classe aluno através de uma interface, representada pela classe *Form* apresentada anteriormente. A classe *Form* se comunica com o serviço Localizar evento da disciplina, através da classe *Location Event*, que por sua vez, herda os atributos e métodos da classe *LBS Service*.

A classe *LBS Service* de acordo com o perfil proposto, herda todos os atributos e métodos da classe *Application Service* e possui as mesmas características e funções explicadas anteriormente. Além disso, a classe *LBS Service* é responsável pela execução de serviços móveis que dependem da localização do usuário móvel. Por esse motivo ela possui os métodos *get_ location_ user()* e *update_ location_ user()* para fazer a busca e atualização da localização do usuário móvel respectivamente.

A classe *Location Event*, possui os dados necessários para obter as informações sobre o serviço Localizar evento da disciplina, os dados dos usuários que podem acessar o serviço, que são validados pelo método *validate_ user()* e a localização do evento da disciplina em relação ao usuário móvel. A partir destes dados, é possível retornar a localização do evento escolhido pelo usuário móvel.

A localização do evento e do usuário móvel é obtida através dos atributos latitude, longitude e área de cobertura (*covering_ area*). A classe *Location Event* também possui mais quatro métodos em especial: O método *get_ data_ event()*, que busca os dados do evento da dsiciplina para o usuário móvel. Depois vem o método *get_ distance_ information()*, que informa a distância da posição do usuário em relação ao evento escolhido da disciplina, com base na localização do evento que é buscada através do método *get_ event_ location()*. Por último temos o método *get_ map*, que exibe o mapa para o usuário móvel da localização do evento em relação a sua posição atual.

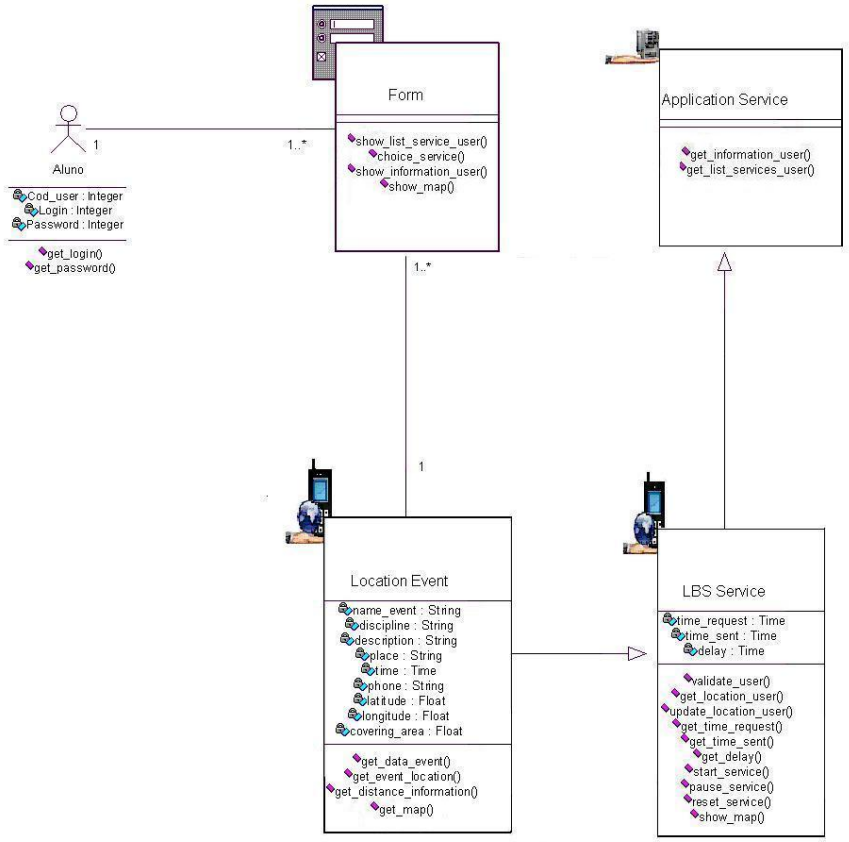


Figura 4.4: Diagrama de classes do Serviço "Localizar evento disciplina"

4.4.2 Diagrama de Sequência

O diagrama de sequência apresentado na Figura 4.5, é um diagrama de interação entre os objetos das classes Aluno, *Form* e *Location Event*, mostrando uma visão temporal das rotinas do caso de uso "Localizar evento disciplina". Este diagrama possui as mesmas características apresentadas anteriormente.

Este diagrama foi feito somente entre as classes Aluno, *Form* e *Location Event*, que representa a associação no diagrama de classes apresentado anteriormente. As outras classes não foram utilizadas pelo mesmo motivo explicado anteriormente.

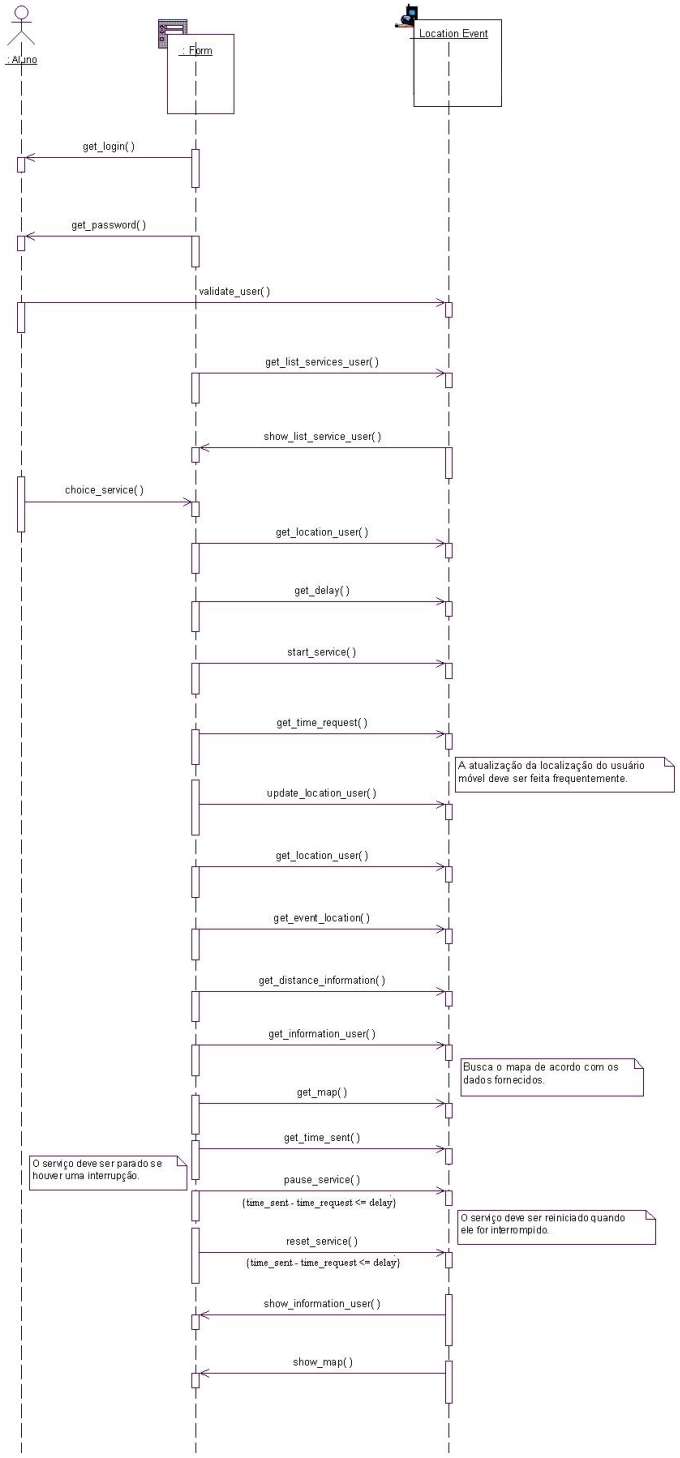


Figura 4.5: Diagrama de sequência do Serviço "Localizar evento disciplina"

4.5 Comparação de modelagem sem a utilização do perfil *Mobile Services*

O diagrama de classes da Figura 4.6, como se pode ver, não seria uma boa solução colocar todos os atributos e métodos dos serviços juntos, pois além da falta de organização dos mesmos, não existe herança dos atributos e métodos como foi feito utilizando o perfil proposto. Além disso, também contraria o princípio da orientação à objetos, onde cada classe deve haver objetos com as mesmas características e funções, o que não acontece devido ao fato de estar agrupando em uma mesma classe serviços móveis e serviços baseados em localização. Dessa maneira, os atributos e métodos ficam misturados dificultando o entendimento do modelo, o que irá dificultar também na hora de implementar.

Figura 4.6: Diagrama de classes do Serviço "Localizar evento disciplina"

Os diagramas de classes e de sequência utilizados neste trabalho poderiam ser usados para fazer a modelagem sem utilizar o perfil proposto. Porém, no diagrama de classes e de sequência apresentados na Figura 4.7 e na Figura 4.8, pode-se perceber que a solução do problema sem a utilização do perfil proposto não resolveria o problema completamente, pois as restrições impostas pela mobilidade e o fato do serviço ser dependente da localização do usuário móvel não seriam tratadas.

Dessa forma, não existe uma descrição detalhada dos serviços, com suas restrições escrita em linguagem formal e OCL. Pode-se perceber isso também no diagrama de sequência, onde não são colocadas as restrições em OCL, devido ao fato de não haver a descrição em linguagem formal e OCL como foi feito no perfil proposto.

O uso do perfil facilitou a modelagem, onde é necessário somente acrescentar o novo serviço utilizando algum estereótipo de serviço, seja do tipo *Mobile Service* ou *LBS Service*, com seus atributos e métodos. Feito isso, basta repetir o uso dos demais serviços Mobile Service e LBS Service, que já foram criados anteriormente. Além disso, com o uso do perfil são tratadas as características e restrições específicas para modelagem de serviços em ambientes móveis, utilizando a descrição dos métodos, descrição das restrições utilizando linguagem formal e OCL e ícones para facilitar a identificação do tipo de serviço, como foi feito na definição do perfil proposto.

O perfil também possui recursos necessários para completar a semântica para a modelagem específica desses tipos de serviços em ambientes móveis, como podemos ver neste trabalho.

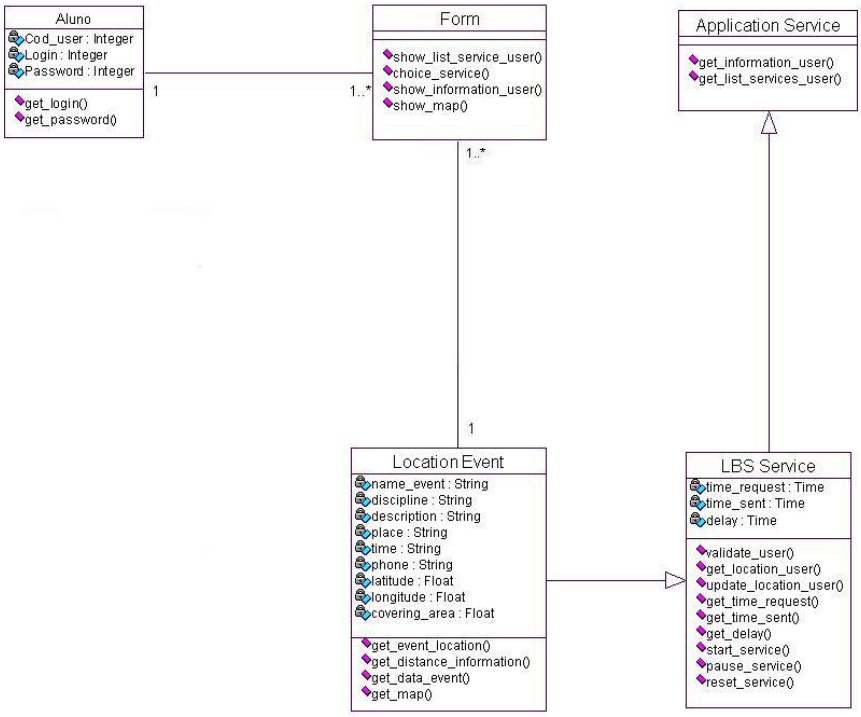


Figura 4.7: Diagrama de classes do Serviço "Localizar evento disciplina"

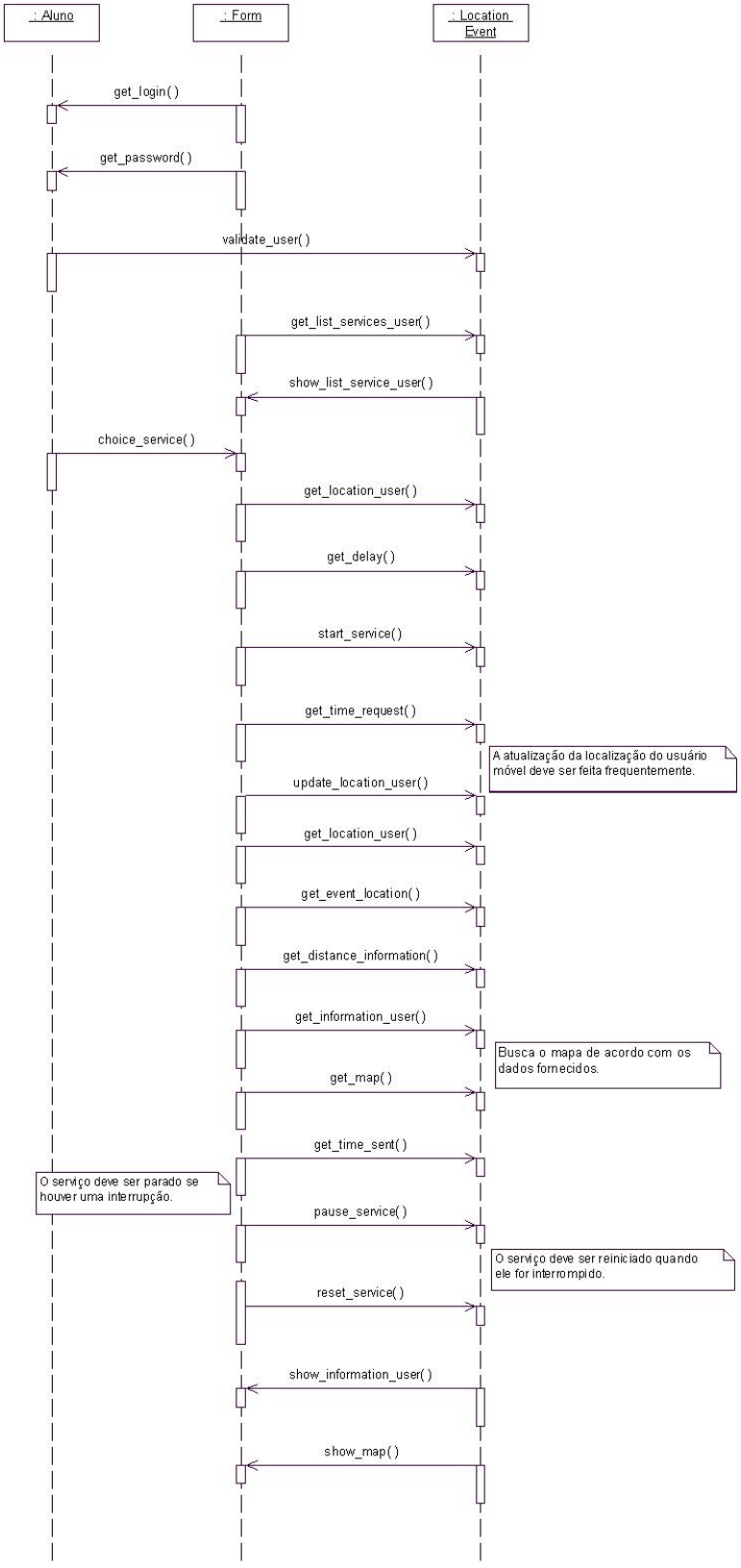


Figura 4.8: Diagrama de sequência do Serviço "Localizar evento disciplina"

Capítulo 5

Resultados e Discussão

5.1 Introdução

Este capítulo apresenta a discussão dos resultados do perfil *Mobile Services* proposto. O perfil foi proposto com o objetivo de modelar serviços em ambientes móveis utilizando a UML para a modelagem desses tipos de sistemas, resolvendo os problemas impostos pela mobilidade.

Este estudo contribui para que os desenvolvedores desses tipos de sistemas possam construir *softwares* de melhor qualidade na fase de projeto e como consequência a redução de custos e o aumento da qualidade.

5.1.1 Resultados e Discussão

Após a utilização do prefil proposto, podemos perceber que os requisitos mencionados na sessão 3.1.1, foram todos alcançados:

- Um diagrama deverá ser capaz de modelar serviços comuns e baseados na localização do usuário móvel, pois foram criados estereótipos de serviços que permitem a modelagem de serviços em ambientes móveis que dependam ou não da localização do usuário móvel.

- Os diagramas deverão ser de fácil compreensão e utilização, devido ao fato dos estereótipos criados serem semelhantes às interfaces de serviços disponibilizadas

em sistemas reais. Dessa forma, basta criar o elemento do modelo, nomeá-lo e definir sua posição e atributos que são opcionais, economizando tempo para fazer a modelagem.

- A representação gráfica (ícones) utilizada nos estereótipos dos diagramas, devem representar os tipos de serviços. Foram criados dois tipos de serviços: comuns e dependentes da localização do usuário móvel, com ícones indicando serviços(mão), mobilidade(celular) e localização(mundo).

- O modelo seguiu o metamodelo UML, sendo de fácil compreensão, para que outras pessoas que não são da área de desenvolvimento de sistemas possam compreender o modelo proposto;

- O modelo é independente de tecnologia ou linguagem de programação, baseado na SOA;

- O modelo utilizou restrições em cada estereótipo de serviços criados, estaticamente ou dinamicamente através de expressões textuais ou OCL quando possível.

Na Figura 4.2, foi visto que sem o uso do estereótipo «*Mobile Service*» não é possível oferecer serviços que dependam da localização do usuário móvel, devido as limitações impostas pela mobilidade, como atualização da localização do usuário móvel. Os estereótipos UML criados permitiram uma solução de fácil compreensão para os desenvolvedores e projetistas desses tipos de sistemas.

Analisando a Figura 4.4, percebe-se que com o perfil proposto, a modelagem de serviços adaptou-se às restrições impostas pela mobilidade, permitindo a modelagem de Serviços Baseados na localização do usuário móvel. Pode-se ver esta característica através do estereótipo «*LBS Service*».

A proposta de extensão de CONALLEN (1999), apesar de ter sido feita para a modelagem de sistemas *Web*, ajudou na análise da proposta do perfil *Mobile Services*, já que sistemas web se baseiam na arquitetura cliente-servidor, semelhante a requisição e prestação de serviços. A mesma arquitetura é aplicada para serviços em ambientes

móveis, com alguns elementos a mais como foi descrito na sessão 2.3.5. As semelhanças estão na troca de mensagens descritas em XML e distribuídas com o uso de protocolos de tranporte da Internet como por exemplo HTTP e TCP/IP. Dessa maneira, um serviço quando necessita de outro serviço, informa os requisitos na mensagem e envia para o serviço receptor que analisa e processa a mensagem e envia uma mensagem para o serviço requisitante, que analisa a resposta e extrai as informações necessárias.

A diferença entre a SOA e a arquitetura baseada em componentes é que os serviços não invocam métodos associados a outros serviços, eles invocam o próprio serviço. Isso acontece através da WSDL, que possui informações da localização dos serviços e os detalhes de sua interface que são descritos em XML. A definição do que cada serviço faz, como eles se comunicam com outros serviços e onde podem ser encontrados são feitos pela WSDL. Dessa forma, foi visto que com o perfil proposto para sistemas Web não é possível modelar serviços em ambientes móveis, devido ao fato dos componentes necessários não serem modelados como serviços e também pelo fato de não ser tratado na modelagem as restrições impostas pela mobilidade, como tempo que a infomação é válida e localização dos usuários móveis. Assim, observa-se que o conjunto de extensões UML propostas no perfil *Mobile Services* apresentado anteriormente, complementa o perfil proposto por CONALLEN (1999). Dessa maneira, a junção dos dois perfis facilita na modelagem de serviços em sistemas *Web*.

O Perfil para Sistemas de Tempo Real proposto em (GRAF, 2005), preocupa-se com o comportamento do sistema em relação ao tempo, onde qualquer falha poderá trazer grandes prejuízos. Esses tipos de sistemas possuem restrições maiores em relação ao tempo. Já nos sistemas móveis o tempo é uma restrição, mas não é necessário um projeto tão cuidadoso quanto nesses tipos de sistemas, dispensando a modelagem por exemplo do processador, redes de comunicação e o relacionamento destes com o *software*. Também é necessário a modelagem do comportamento do sistema em relação a algum tipo de evento que pode ocorrer e também quanto a qualidade de serviços(QoS), permitindo obter os conceitos, exigências e propriedades de qualidades de serviços e suporta qualquer análise específica sobre segurança de sistemas. As outras questões não se aplicam na modelagem de serviços em ambientes móveis, devido às

grandes exigências que estes sistemas exigem.

O perfil proposto complementa outros trabalhos como: Perfil para Sistemas Móveis, Sistemas Distribuídos Móveis e modelagem para SMS *(Simple Mobile Services)* citados em (GRASSI, 2004), (SIMONS, 2007) e (BROLL, 2007), respectivamente, que tratam de questões referentes à mobilidade, permitindo que seja feita a modelagem de serviços para ambientes móveis.

Por fim, foram encontrados alguns trabalhos sobre Perfis UML para Arquiteturas Orientadas a Serviços e *Web Services*, citados em (HECKEL, 2003), (ORTIZ, 2006), respectivamente. Os dois Perfis UML tratam de questões como interoperabilidade entre as aplicações, reusabilidade e plataformas independentes. Estas propostas de Perfis UML não levam em consideração a prática de serviços em ambientes móveis.

Os benefícios esperados com o uso do perfil proposto foram: Facilidade de manutenção e na compreensão do código, pois os componentes de serviços ficam separados em módulos independentes, não necessitando a procura de trechos de código por todas as classes de aplicação, diminuindo o tempo para a manutenção. A compreensão do código é pelo mesmo motivo anterior, pois como os serviços ficam separados em módulos, faz com que o desenvolvedor se concentre em poucos módulos para entender a aplicação.

Outro ponto foi a limitação do desempenho por causa do número de interfaces dos serviços. Devido ao fato do código ser bem modularizado, adicionar novos serviços se torna mais fácil. Por exemplo, adicionar um novo serviço, que utiliza a tecnologia *Web Services*. Para isso, cria-se o novo serviço desejado e informa ao *Web service* que existe mais um serviço. Por fim, basta informar o que esse serviço faz, como é feita a comunicação com esse serviço e a sua localização.

Dessa maneira, foi observado que o perfil proposto aumentou a facilidade para a geração de *plugins*, devido aos componentes de serviços já estarem prontos, gerando o código de maneira mais rápida e mais fácil de se entender. Essa características torna os sistemas mais flexíveis, além de trazer os benefícios da mobilidade e do baixo custo.

Algumas limitações apresentadas no ambiente móvel que não foram tratadas neste perfil podendo ser feito um estudo posteriormente sobre o assunto são: limi-

tações de performance, limitação do tamanho das aplicações e bibliotecas de funções reduzidas. Outra questão é que o perfil proposto não trata da reutilização de código de maneira automática, onde seria referenciado a parte do código escolhida ao invés de replicar as informações já modeladas no modelo de domínio da aplicação. Os conceitos do domínio são modelados de forma clara e a relação das interfaces com o usuário com esses conceitos pode ser perfeitamente visualisado, facilitando o entendimento das regras de negócio do sistema.

Capítulo 6

Conclusões

Este trabalho possibilitou uma visão geral dos conceitos de computação móvel, suas principais aplicações possibilitando perceber seus problemas e limitações que devem ser levados em conta para que sejam minimizados cada vez mais. Foram mostradas também as principais aplicações e serviços da computação móvel, onde se pôde ver a importância da computação móvel no nosso cotidiano.

Os serviços estão sendo bastante usados hoje por várias empresas, principalmente pelas empresas de telefonia celular, possibilitando o aumento de suas aplicações e melhoria dos serviços prestados para atender de forma satisfatória os usuários. Para que os problemas encontrados no desenvolvimento de sistemas, como a má especificação dos requisitos do sistema, prazos de entrega ultrapassados, sejam minimizados, estes requisitos devem ser bem visualizados pelo desenvolvedor através da modelagem do sistema. Dessa maneira, foi constatado que com o uso do perfil UML *Mobile Services* proposto é possível a modelagem de serviços que dependem ou não da localização do usuário móvel. Além disso, o perfil facilita a modelagem de serviços em ambientes móveis de acordo com suas respectivas propriedades, métodos e restrições de cada estereótipo.

Com o Perfil UML proposto, pode-se perceber que não basta basear somente nos elementos definidos pela UML. É preciso buscar extensões que tratam da modelagem de negócio. O perfil proposto possui as seguintes características: maior reusabilidade, independente de tecnologia e facilidade de composição entre classes da

aplicação. Deve ser levado em conta os problemas que ocorrem nesse tipo de aplicação, limitações de performance, limitação do tamanho das aplicações e bibliotecas de funções reduzidas. Outra questão é que o perfil proposto não trata da reutilização de código de maneira automática, onde seria referenciado a parte do código escolhida ao invés de replicar as informações já modeladas no modelo de domínio da aplicação. Os conceitos do domínio são modelados de forma clara e a relação das interfaces com o usuário com esses conceitos pode ser perfeitamente visualisado, facilitando o entendimento das regras de negócio do sistema.

A elaboração de um modelo a partir dos conceitos apresentados no perfil proposto possibilita a construção do sistema de forma mais rápida e reduz o número de falhas durante a fase de projeto. A modelagem de serviços em ambientes móveis feita neste trabalho utilizou como base as restrições impostas pelos serviços (regras de negócio) e pela mobilidade.

A tecnologia *Webservices*, que é baseada na SOA, facilitou a modelagem do desenvolvimento de serviços para ambientes móveis, deixando claro a facilidade de incluir novos serviços do mesmo domínio. A comunicação entre os serviços é de forma padronizada, independente de plataforma e de linguagem de programação. Isso acontece devido o padrão ser descrito em XML, o qual pode gerar documentos complexos. Por exemplo, um sistema desenvolvido em Delphi e rodando em um servidor Microsoft pode acessar, um serviço feito em Java rodando em um servidor Linux. Além disso, diminuiu o trabalho dos desenvolvedores desses tipos de sistemas, devido ao fato dos componentes estarem prontos e poderem ser inseridos em uma ferramenta CASE para geração automática de código, gerando o serviço desejado.

6.1 Trabalhos Futuros

Algumas propostas de trabalhos futuros são:

- Avaliar a utilização do perfil *Mobile Services* durante a implementação de uma aplicação real utilizando a tecnologia *Web Services*;
- Especificação e avaliação de um modelo semelhante a este utilizando a linguagem MOF.
- Utilização do Serviço de Registro UDDI (*Universal Description, Discovery and Integration*) para obtenção da descrição dos serviços providos e consumidos pelos componentes;
- Fazer testes em um sistema real com uma ferramenta CASE e avaliar o modelo para que siga corretamente o padrão estabelecido pelo OMG.
- Construção de um *framework* para serviços em ambientes móveis modelados neste perfil. Para geração de Clientes e Servidores de Serviços que serão validados a partir dos atributos, métodos e restrições de cada estereótipo.
- Avaliar e testar as expressões de restrições escritas em linguagem OCL através de uma ferramenta CASE que suporte restrições escritas nesta linguagem.

Referências Bibliográficas

ALONSO, G.;CASATI, F.; KUNO, H.; MACHIRAJU, V. **Web Services: Concepts, Architecture and Applications**. New York: Springer Verlag, 2004.

AMJAD, Umar. **Mobile Computing and Wireless Communications**. Nge Solutions. Jul. 2004.

BMP. **Bussines Process Management Institute**. Disponível em: <http://bpmi.org>. Acesso em: 15 dez. 2007.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **Unified Modeling Language User Guide**. 2.ed. Boston: Addison-Wesley Professional, 2005.

BROLL, Gregor; HUBMANN, Heinrich; PREZERAKOS, George N.; KAPITSAKI, Georgia; SALSANO, S. **Modeling Context Information for Realizing Simple Mobile Services Mobile and Wireless Communications**. Budapest, Germany, 2007.

CONALLEN, Jim **Building Web Applications with UML**. Addison-Wesley Longman, 1999.

ERIKSON, Hans; PENKER, M. **UML 2 Toolkit**. 2.ed. New York: John Wiley, 2004.

FERNANDÉZ, Lidia F.; MORENO, Antônio V. **An Introduction to UML Profiles**, v. 5. Málaga, Spain: The European Journal for the Informatics Professional, 2004.

FOWLER, Martin. **UML Distilled: A Brief Guide to the Standard Object Modeling Language**. 3.ed. Boston: Addison-Wesley Professional, 2003.

GILMORE, S.; HAENEL, V.; HILLSTON, J; TENZER, J. **A design environment for mobile applications**. In: Parallel and Distributed Processing Symposium, Scotland: The University of Edinburgh, 2006.

GRAF, Susanne; OBER, Ileana; OBER, Iulian **A Real-Time Profile for UML**. France: Verimag-Centre Equation, 2005.

GRASSI, Vincenzo; MIRANDOLA, Raffaela; SABETTA, Antonio **A UML Profile to Model Mobile Systems**. In: International conference on the unified modeling language: Italy: University di Roma Tor Vergata, 2004.

HECKEL, Reiko; LOHMANN, Marc; THONE, S. **Towards a UML Profile for Service-Oriented Architectures**. University of Paderborn, Germany, 2003.

JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James **Unified Modeling Language Reference Manual**. 2.ed. Reading, Mass: Addison-Wesley Professional, 2004.

MACHADO, J. C. **Um estudo sobre o desenvolvimento orientado a serviços**. Dissertação de Mestrado - PUC,RJ, Rio de Janeiro, 2004.

MATEUS, Geraldo R.; LOUREIRO, Antônio A. F. **Introdução à Computação Móvel**. 11a Escola de Computação, Rio de Janeiro, 2004.

MCCOY, David W.; MCCOY, Yefim V. **Service-Oriented Architecture: Mainstream Straight Ahead**. Disponível em: <http://www.gartner.com/pages/story.php.id.3586.s.8.jspbpmi.org>. Acesso em: 15 nov. 2007.

NIELSEN, Elena Sanchez; RUIZ, Sandra Martin; PEDRIANES, Jorge Rodriguez **Mobile and Wireless Networks**, V.263. Proceedings of the 6th International Conference on Web Engineering, Palo Alto, California, 2006.

OASIS, R. **Reference Model for Service Oriented Architecture 1.0**. Disponível em: <http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf>. Acesso em: 20 nov. 2007.

OCL (Object Constraint Language) **UML 2.0 specification**. Disponível em: <http://www.omg.org/cgi-bin/doc?ptc/2003-10-14>. Acesso em: 05 dez. 2007.

ORTIZ, G.; HERNANDEZ, J. **Toward UML Profiles for Web Services and their Extra-Functional Properties**, V.18. ICWS '06. International Conference, Chicago, 2006.

PAULA FILHO, Wilson P. **Engenharia de Software: Fundamentos, Métodos e Padrões**. 2.ed. Rio de Janeiro: LTC, 2003.

RUMBAUGH, James R.; BLAHA, M. R. **Object-Oriented Modeling and Design with UML**. 2.ed. Reading, Mass, Prentice Hall, 2004.

SCHILLER, Jochen; VOISARD, A. **Location-Based Services**. Morgan Kaufmann Publishes is an imprint of Elsevier: San Francisco, 2004.

SILVA, Alberto, V. C. **UML Metodologias e Ferramentas CASE**. Centro Atlântico, Portugal, 2005.

SIMONS, C.; WIRTZ, G. **Modeling context in mobile distributed systems with the UML**, V.18. Academic Press, Orlando, FL, 2007.

SOMMERVILLE, Iam **Software Engineering**. 8.ed. Pearson Addison Wesley, 2007.

STREET, Julie; GOMAA, H. **Software Architectural Reuse Issues in Service-Oriented Architectures**, V.18. IEEE Computer Society, Washington, DC, 2008.

TSALGATIDOU, Aphrodite; VEIJALAINEN, Jari; MARKKULA, Jouni, KATASONOV, Artem; HADJIEFTHYMIADES, Stathes **Mobile E-Commerce and Location-Based Services: Technology and Requirements**. 2004.

ZENI, Cristine; BORSATO, Emerson P.; PINTO, José S.; MALAFAIA, Oswaldo **Panorama do uso de Computação Móvel com conexão Wireless**. Universidade Federal do Paraná, Curitiba, PR, 2006.