UNIVERSIDADE FEDERAL DE PELOTAS
INSTITUTO DE FÍSICA E MATEMÁTICA
CURSO DE BACHARELADO EM INFORMÁTICA

Um *Framework* para a Construção de Sistemas de Monitoração, Alerta e Intervenção Remotos

por

WAGNER LUÍS CARDOZO GOMES DE FREITAS

Monografia apresentada ao Curso de Bacharelado em Informática do Instituto de Física e Matemática da Universidade Federal de Pelotas, como requisito parcial para obtenção do título de Bacharel em Informática.

Prof. Maurício Nunes Porto
Orientador

Prof. Cláudio Vianna Villela
Bel. Roger Al-Alam Krolow
Co-orientadores

Pelotas, janeiro de 2001

"A razão cardeal de toda a superioridade humana é sem dúvida a vontade. O poder nasce do querer. Sempre que o homem aplique a veemência e a perseverante energia de sua alma a um fim, ela vencerá os obstáculos, e se não atingir o alvo, fará pelo menos coisas admiráveis."

– José Martiniano de Alencar

Agradecimentos

Na elaboração deste trabalho agradeço, inicialmente, ao professor Maurício Nunes Porto, orientador, que soube reportar minhas indagações com a presteza e ponderação necessárias, demonstrando seu grande saber com a paciência de um sábio e a tolerância de um pai, permitindo que as idéias afigurassem a imagem pretendida, dando corpo lógico e conclusão à objetivação inicial.

Agradeço ao professor Gil Carlos Rodrigues Medeiros que transmitiu os conhecimentos inicias à realização deste trabalho, e que pela sua magistral personalidade soube a cada gesto transmitir sua sabedoria, mostrando que o norte de nossa realização está na seriedade de nosso empreendimento e na responsabilidade de buscar sempre o melhor.

Aos demais professores da Universidade Federal de Pelotas, especialmente aos do curso de Informática, os quais sem distinção mostraram-se capazes nas suas atribuições, transmitindo os seus conhecimentos de maneira acessível.

A meus pais, que com seu amor e respeito permitiram, com total liberdade, que formasse meus conhecimentos seguindo meus desejos, incentivando todas as minhas realizações, mesmo as mais pequenas, como se todas elas fossem o caminho necessário a minha glorificação.

Agradeço, por fim, a quem sempre deverá ser o Primeiro, ao meu Pai Celeste, cuja vontade permitiu que concluísse este trabalho após o acidente automobilístico que sofri, restabelecendo-me a tempo certo. A Jesus Cristo nosso sagrado Senhor, minha completa devoção.

Sumário

Lista de Abreviaturas

AMPS – Advanced Mobile Phone System
BSS – Base Station System
CDMA – Code Division Multiple Access
CDPD – Cellular Digital Packet Data
CPU – Central Process Unit
DTD – Document Type Definition
GSM – Global System for Mobile Communications
HDML – Handheld Device Markup Language
HLR – Home Location Register
HTML – Hypertext Markup Language
HTTP – Hypertext Transfer Protocol
ICMP – Internet Control Message Protocol
IMTS – Improved Mobile Telephone System
IP – Internet Protocol
JDC – Japanese Digital Cellular
MAP – Mobile Application Part
MO-SM – Mobile-Original Short Message
MS – Mobile Station
MSC – Mobile Switching Center
MT-SM – Mobile-Terminated Short Message
MTSO – Mobile Telephone Switching Office
OSI – Open Systems Interconnection
PAP – Push Access Protocol
PCN – Personal Communications Network
PCS – Personal Communications Service
PDA – Personal Digital Assistant
PDU – Protocol Data Unit
PPG – Push Proxy Gateway
RAM – Random Access Memory
RFC – Request For Comments
ROM – Read Only Memory
SGML – Standart Generalized Markup Language
SIM – Subscriber Identify Module
SMD-PP – Short Message Delivery Point-to-Point
SME – Short Messaging Entity
SMS – Short Message Service
SMSC – Short Message Service Center
SMS—GMSC – SMS-Gateway/interworking Mobile Switching Center
SMTP – Simple Mail Transfer Protocol
SNMP – Simple Network Manegement Protocol
SS7 – Signaling System 7
SSL – Secure Sockets Layer
TCP – Transmission Control Protocol
TCP/IP – Transmission Control Protocol/Internet Protocol
TDMA – Time Division Multiple Access
TLS – Transport Layer Security

UDP – User Datagram Protocol
UML – Unified Modeling Language
URI – Universal Resource Identifier
URL – Uniform Resource Locator
USSD – Unstructured Supplementary Service Data
VLR – Visitor Location Register
WAE – Wireless Application Environment
WAP – Wireless Application Protocol
WBMP – Wireless Bitmap
WCMP – Wireless Control Message Protocol
WDP – Wireless Datagram Protocol
WML – Wireless Markup Language
WMLScript – Wireless Markup Language Script
WSP – Wireless Session Protocol
WSPB – Wireless Session Protocol Browsing
WTA – Wireless Telephony Application
WTAI – Wireless Telephony Application Interface
WTLS – Wireless Transport Layer Security
WTP – Wireless Transaction Protocol
WWW – World Wide Web
XML – Extensible Markup Language

Lista de Figuras

Lista de Tabelas

Resumo

A tecnologia das comunicações vem experimentando uma rápida evolução que, além de apresentar novos meios e dispositivos, a estes acrescenta diversos recursos úteis e inovadores. Entre outros acréscimos dessa evolução, a comunicação de dados sem fio e a integração das redes de comunicação com a rede de computadores Internet, tem propiciado o surgimento de novas aplicações e formas de uso desses recursos.

Este trabalho descreve inicialmente, a base tecnológica atual da comunicação de dados sem fio, para em seguida apresentar um modelo genérico de sistema computacional que é resultado da aplicação dessas novas tecnologias de comunicação integradas a sistemas de computação.

Esse modelo resultante constitui um *framework* para a construção de sistemas voltados à monitoração de ambientes, com a capacidade de emitir mensagens de alerta para dispositivos móveis e permitir que os usuários intervenham remotamente sobre o ambiente monitorado, através do mesmo dispositivo móvel.

Esse *framework* está baseado no ambiente e linguagem Java, tendo sido construído segundo o paradigma de orientação a objetos.

Abstract

The communication's technology comes trying fast evolutions that, besides presenting new means and devices, to theses increase several useful and innovative resources.

Among other increments of that evolution, the wireless networks of communication and the integration of that communication networks with the Internet, has been propitiating the appearance of new applications and forms of use of those resources.

Initially, this work describes the current technological base of the wireless communication, for after that to presents a generic model of computational system that is resulting of the application of those new communication technologies integrated with computer systems.

That resulting model makes up a framework for the construction of systems intended to the monitoring of environments, with the ability to emit alert messages for mobile devices and to allow the users to intervene remotely on the monitored environment, through the same mobile device.

The framework is based on the environment and language Java, having been built according to the object-oriented paradigm.

1 Introdução

O homem, no uso racional da ciência, vence as distâncias, permitindo-se utilizar a tecnologia para alcançar qualquer parte do espaço através da comunicação. Os dispositivos móveis e os sistemas computacionais com acesso a Internet trazem a globalização entre os povos e permite que através de dados se transmitam fatos, a conversação e a imagem em tempo real, por exemplo.

O uso da rede Internet cresce a cada dia. A utilização da Internet como rede de comunicação entre dispositivos móveis e sistemas diversos ainda desabrocha, daí a necessidade de serem realizados estudos e a busca de conhecimentos nessa área.

Os telefones celulares, que utilizam as redes de comunicação sem fio, distribuíram-se por toda a sociedade, e hoje são vistos como o instrumento da comunicação do presente e cada vez mais do futuro, em vista das novas aplicações e funcionalidades disponíveis para este dispositivo.

A integração entre as redes de telefonia celular digital e a Internet permite que novas possibilidades de uso surjam constantemente. Entre as aplicações já existentes, onde o usuário de um telefone celular digital pode tomar a iniciativa, figuram a possibilidade de efetuar pagamentos sem precisar transportar dinheiro, comandar a ativação ou desativação de luzes, aparelhos de ar condicionado e outros equipamentos eletrodomésticos de sua residência ou até intervir remotamente em atividades médicas, o que tem sido caracterizado como uma novíssima área denominada "telemedicina" [REG 00].

Outras aplicações oriundas dessa integração caracterizam-se pela iniciativa extra-usuário, quando este recebe notificações que contém apenas material informativo ou mensagens com pedidos de resposta ou intervenção. Entre tais aplicações, existem atualmente serviços de informações financeiras, que informam sobre saldos e cotações, com o aviso imediato em caso de alta ou queda de ações especificadas pelo usuário, bem como sistemas de monitoramento remoto de redes de computadores, que avisam os administradores quando algum equipamento apresentou alguma falha ou há congestionamento na rede, por exemplo [YUR 00].

Certamente é possível e desejável ter-se aplicações que integrem os dois tipos de iniciativa, ou seja, partindo do usuário ou partindo do sistema remoto. Tais aplicações podem emitir avisos ou alertas aos usuários, permitindo que estes respondam com comandos

ou ações sobre os sistemas remotos; possibilitam também que os próprios usuários tomem as iniciativas dessas operações.

As áreas de aplicação de um sistema desse tipo, que faça a monitoração, o alerta imediato e permita intervenções remotamente, são muito vastas. Cada ambiente ou sistema a ser monitorado e gerido possui suas particularidades, diferindo uns dos outros em alguns detalhes que envolvem a forma de monitoração e as possibilidades de intervenção. Mesmo assim, têm muita coisa em comum: a necessidade de fazer com que os alertas cheguem rapidamente aos usuários, a infraestrutura de comunicação e interface que permita a intervenção remota e o registro das ocorrências importantes, entre outros aspectos.

Identificar e reunir esses aspectos comuns entre esses sistemas permite que seja construída uma infraestrutura de *software* que forneça os meios de implementar e selecionar os mecanismos adequados, necessários a cada aplicação específica.

Na terminologia da orientação a objetos, esse requisito poder ser suportado por um *framework* orientado a objetos, que é um gabarito que descreve os aspectos comuns e torna-os flexíveis o bastante para adequar o sistema construído aos aspectos particulares de cada caso.

Neste trabalho é apresentado um *framework* de apoio à construção de sistemas de monitoração, alerta e intervenção remotos, baseado no uso de tecnologias adotadas pela telefonia celular digital (SMS e WAP), servidores *Web* e no ambiente Java.

Este trabalho está dividido em 6 capítulos. O capítulo 2 descreve a evolução dos dispositivos sem fio, abordando como assunto principal as tecnologias como *pager* e telefone celular. Os capítulos 3 e 4 apresentam as tecnologias de comunicações via telefonia celular digital adotadas, respectivamente SMS e WAP. O capítulo 5 apresenta e discute o *framework* desenvolvido. Encerrando esta monografia, o capítulo de conclusões tece algumas considerações a respeito do trabalho, descrevendo o estágio atual de desenvolvimento do *framework* e suas necessidades de continuidade.

2 Comunicação de Dados Sem Fio

Nos últimos anos tem crescido o grupo de usuários que necessitam comunicar-se estando em constante mobilidade. Essa necessidade não pode ser suprida pelo sistema telefônico tradicional, e conseqüentemente, existe uma tendência cada vez maior de se utilizar sistemas que empregam ondas de rádio em vez de fios e fibras para comunicação [TAN 97].

Os sistemas de comunicação móvel desempenham um papel cada vez mais importante na ligação de *notebooks*, telefones de bolso e agendas eletrônicas em rede, tornando disponíveis recursos como correio eletrônico e pesquisa de informações em bancos de dados remotos instalados em qualquer parte do mundo. Problemas como incompatibilidade de produtos e serviços e a grande diferença de tecnologia disponível em cada país representa um desafio na padronização desses sistemas e uma dificuldade a mais na redução de custos e integração de redes de comunicação.

A seguir, é apresentada uma resenha histórica da evolução tecnológica dessa área, relacionando as tecnologias em ordem cronológica. Nesse sentido, são descritos o sistema *paging* e o telefone celular, este último dispositivo de efetiva importância na comunicação atual.

2.1 Serviço de *Paging*

Um sistema de *paging* atual tem por objetivo fazer a entrega, via ondas de rádio, de pequenas mensagens de texto.

Os primeiros sistemas de *paging* funcionavam através do uso de alto-falantes espalhados por um ambiente como, por exemplo, um hospital. Quando era necessário enviar um recado de forma rápida a um médico, a informação era propagada por estes alto-falantes de forma que, em um pequeno espaço de tempo, a mensagem, geralmente curta, chegasse ao destinatário [TAN 97].

Posteriormente, ocorreu uma evolução destes sistemas e a informação passou a ser distribuída através de pequenos dispositivos com mostradores onde as mensagens são exibidas, os quais tem a designação de *pager*.

Para realizar a comunicação com este dispositivo, a pessoa efetua uma ligação para a empresa provedora do serviço, fornece um código de segurança, o número do *pager* e a pequena mensagem a ser enviada.

Após, estas informações são recebidas por um computador e transmitidas até uma antena. Caso o *pager* esteja localizado na área de alcance dessa antena, a informação é enviada diretamente, via ondas de rádio, ao dispositivo do usuário. Senão, a informação pode ser repassada a outros meios de comunicação, como um satélite, antes de chegar ao destino. Geralmente, quando a mensagem é recebida pelo equipamento, um som é emitido para alertar o usuário.

Além da possibilidade de enviar recados a um determinado usuário de *pager*, os sistemas de *paging* também possibilitam que uma mesma mensagem seja enviada para vários destinatários.

Com o crescimento do número de usuários, esses sistemas voltaram a ser incrementados e tornaram disponível o envio de mensagens mais longas e a possibilidade do envio de mensagens através de outras formas como, por exemplo, através de um formulário HTML (*Hypertext Markup Language*) disponível na Internet.

Por ser um sistema unidirecional (somente o computador transmite as mensagens) e apenas enviar mensagens de texto, a largura de banda necessária para estes sistemas é pequena, o que o torna barato e simples em comparação com a comunicação através de telefones celulares, por exemplo.

Hoje em dia, ainda se vê uma grande evolução dessa tecnologia, com o surgimento de dispositivos que permitem a comunicação nos dois sentidos (o usuário, além de receber as mensagens, pode enviá-las a partir do próprio *pager*) e o acréscimo de funcionalidade ao sistema pela adição de serviços como agenda pessoal e comunicação com computadores e outros *pagers*.

2.2 Telefone Celular Analógico

Inicialmente, os rádio-telefones móveis utilizavam um único canal para transmissão e recepção, o que exigia que o usuário pressionasse um botão toda vez que quisesse transmitir sua voz. Ou seja, quando o usuário queria conversar, este apertava um botão que ativava o transmissor e desativava o receptor. Além disso, esses sistemas utilizavam um único transmissor, que geralmente era localizado no topo de um edifício alto.

Logo após, surgiu o padrão IMTS (*Improved Mobile Telephone System*), [SCI 98] que também utilizava um transmissor de alta potência localizado em um lugar alto, mas fazia uso de duas freqüências para comunicação, uma para transmissão e outra para

recepção. Isso possibilitou que duas pessoas pudessem se ouvir simultaneamente e que o botão fosse extinto.

O padrão IMTS suportava um número pequeno de canais, o que fazia com que os usuários tivessem que esperar um longo tempo para conseguir um tom de discagem.

Devido à alta potência dos transmissores e do pequeno número de canais disponíveis, esse sistema comprovou ser de limitada capacidade e rapidamente deixou de ser utilizado.

2.2.1 O Sistema AMPS (*Advanced Mobile Phone System*)

Com o surgimento da tecnologia AMPS (*Advanced Mobile Phone System*) [FAR 99] os problemas encontrados nos sistemas IMTS foram reduzidos. No padrão AMPS uma região geográfica é dividida em células (fig. 2.1), com cada uma utilizando um conjunto de freqüências. Com o uso de células relativamente pequenas e o reaproveitamento do conjunto de freqüências de transmissão em células próximas (mas não adjacentes), esse sistema possibilitou uma maior capacidade de chamadas que o padrão IMTS. Além disso, com o uso de células relativamente pequenas, os dispositivos móveis puderam ter seu tamanho reduzido e a potência dos transmissores também pode ser diminuída.

O princípio de reutilização de freqüências baseia-se na divisão de uma determinada área geográfica em células geralmente circulares (para facilitar a visualização, na figura as células são representadas por hexágonos) e de mesmo tamanho. Elas são divididas em grupos de sete unidades com cada uma possuindo um conjunto de freqüências distintas. Desta forma, cada conjunto de freqüências só é reutilizado a uma distância de aproximadamente duas células [TAN 97].

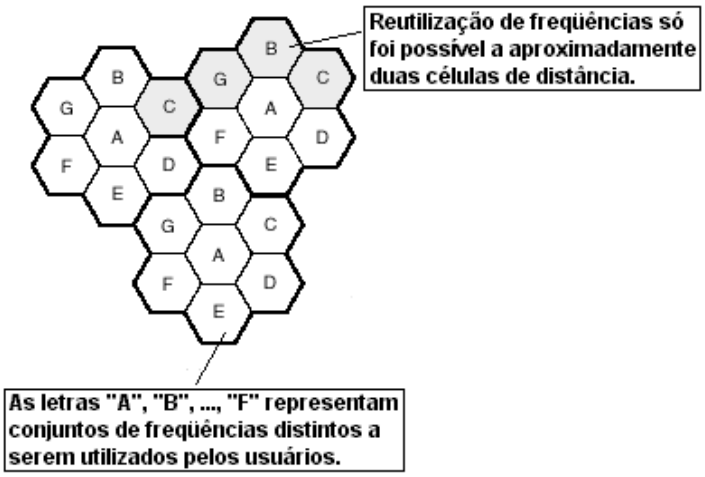

Figura 2.1 – Disposição geográfica de células

Este sistema, assim como o padrão IMTS, necessita que as antenas das estações bases estejam localizadas em lugares altos.

No centro da célula existe uma estação base para onde transmitem todos os telefones da célula (fig. 2.2), sendo a estação base composta por um computador e um transmissor/receptor ligado a uma antena.

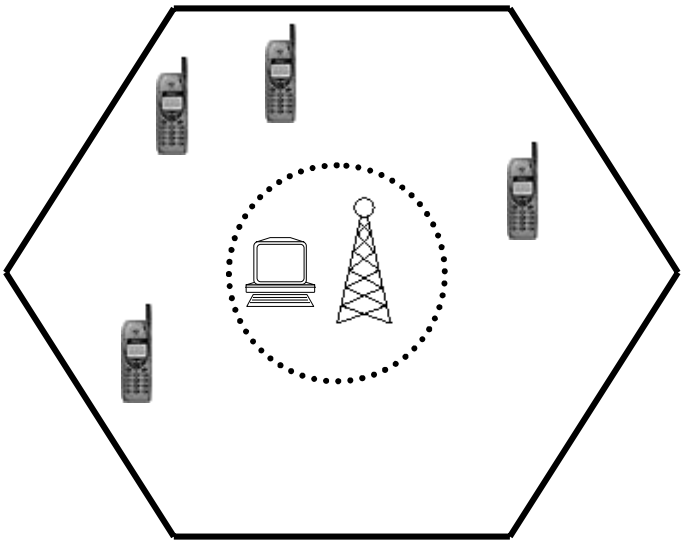

Figura 2.2 – Composição de uma célula

Em uma área onde o número de usuários cresce o suficiente para que o sistema se torne sobrecarregado, a potência da estação base é reduzida e a célula subdividida para que seja possível uma maior reutilização das freqüências.

Em um sistema pequeno tem-se todas as estações bases ligadas a um único posto MTSO (*Mobile Telephone Switching Office*) ou central de chaveamento MSC (*Mobile Switching Center*). Já em um sistema maior, existem vários postos MTSO ligados a um outro MTSO de segundo nível.

A cada instante um telefone móvel ocupa uma célula específica e está sob controle de uma única estação base dessa célula. Quando este telefone deixa uma célula, a estação base identifica que o sinal está enfraquecendo e então realiza uma busca para verificar se o dispositivo móvel se encontra nas estações bases vizinhas. Assim que identifica a localização do telefone, transfere-o para a célula que está com o sinal mais forte e informa o dispositivo sobre a nova estação base que o controla. Se no momento em que ocorre essa transferência há uma chamada em andamento, é solicitado ao dispositivo que troque para outro canal, devido ao fato da não reutilização de freqüências em células adjacentes. Toda

esta operação de transferência consome um tempo bastante pequeno, na ordem de 300 milisegundos [TAN 97].

O sistema AMPS utiliza 832 canais *simplex* agrupados em pares, resultando em 416 canais *full-duplex*. Estes canais são divididos em categorias:

- Controle (unidirecional, da base para a unidade móvel) – gerencia o sistema;
- *Paging* (unidirecional, da base para a unidade móvel) – alerta o dispositivo móvel para entrarem em contato com a central;
- Acesso (bidirecional) – estabelece a chamada e atribui os canais;
- Dados (bidirecional) – utilizado para transferir informações de voz, dados e fax.

Devido à não reutilização de canais em células vizinhas o número de canais de voz disponíveis em cada célula é da ordem de 45 canais [TAN 97].

A intervalos de tempo regulares o telefone móvel varre uma lista de canais de controle a fim de verificar qual apresenta o sinal mais forte no momento. Com a informação do canal de controle, são obtidas as listas dos canais de acesso e *paging*. Feito isso, são enviados o número de série do aparelho e o número do telefone. Isto é realizado várias vezes a fim de identificar qualquer erro. Quando essa informação é recebida pela estação base, esta informa ao posto MTSO sobre a localização atual do usuário.

No momento que o usuário envia um número de telefone pelo teclado, para realizar uma chamada, é transmitida também a informação do número de série para acessar um dos canais de acesso. Quando é obtida a solicitação, caso não tenha ocorrido nenhuma colisão, a estação base informa o posto MTSO. Se a pessoa que está fazendo a chamada for um cliente da empresa do MTSO, este procura por um canal inativo para completar a chamada. Por fim, o telefone móvel conecta-se ao canal de voz selecionado e aguarda até que a outra pessoa atenda ao telefone.

Os telefones inativos ouvem continuamente o canal de *paging* com o objetivo de detectar a ocorrência de chamadas entrantes. Quando é realizada uma chamada para um telefone móvel (via telefone fixo, ou móvel), um pacote é enviado para o posto MTSO local da pessoa chamada a fim de ser localizada. Em seguida essa informação é enviada para a estação base que coordena a célula em que o telefone móvel está localizado. Por fim, o canal de *paging* é utilizado para ver se o dispositivo móvel está ativo e, em caso

afirmativo, é informado o canal de voz onde o telefone móvel deve conectar-se para dar início a comunicação.

2.2.2 Aspectos de Segurança

A tecnologia utilizada no telefone celular analógico é bastante insegura, o que permitiu o surgimento de inúmeros casos de roubos *on-line*. Com a utilização de um receptor de rádio especial que trabalhe na faixa de freqüências utilizadas pelas células e um computador, pode-se fazer uma varredura nos canais de controle e obter informações sobre o número de série dos telefones. Além disso, com o uso do receptor de rádio é possível sintonizar sobre as freqüências dos canais de dados e ouvir a comunicação entre duas pessoas, tornando bastante inseguro a passagem de informações sigilosas através do dispositivo.

2.3 Telefone Celular Digital

Após a primeira geração de telefones móveis (geração dos telefones celulares analógicos), surgiram os telefones celulares digitais que dentre as inúmeras vantagens oferecidas em relação à tecnologia anterior, melhoravam os aspectos de segurança.

Nos EUA, a geração de telefones celulares analógicos era formada apenas pelo padrão AMPS. Quando surgiu a tecnologia digital surgiram vários padrões, e destes, dois se destacaram no mercado: o formado pelos padrões IS-54 e IS-135 e o IS-95. O padrão IS-95 era completamente novo e não apresentava compatibilidade com o padrão AMPS, ao contrário do IS-54 que apresenta um modo duplo de comunicação (analógico e digital) e utiliza os mesmos canais do AMPS.

Na Europa, devido à existência de inúmeros padrões para a telefonia celular analógica, o desejo era o de criar um único padrão para os dispositivos digitais e permitir que, por exemplo, um usuário da França pudesse utilizar seu equipamento na Inglaterra [TAN 97]. Dessa necessidade surgiu o padrão GSM (*Global Systems for Mobile Communications*) que hoje é considerado o padrão que introduzirá as primeiras comunicações através de dispositivos móveis de terceira geração (3G) [IEC 00c, McG 00, 3GM 00]. Várias foram as inovações trazidas por esse padrão, como a utilização de cartões inteligentes onde ficam armazenados o número de série e o número do telefone para que, em caso de roubo, o telefone não pudesse ser reutilizado, além da utilização de técnicas de criptografia nas comunicações. O padrão GSM é um padrão totalmente digital e não apresenta nenhuma garantia de compatibilidade com sistemas analógicos.

O Japão adotou outro padrão, distinto dos demais, denominado JDC (*Japanese Digital Cellular*), que foi criado para ser compatível com o padrão analógico daquele país.

São inúmeros os benefícios da transmissão digital, entre os quais destacam-se:

- Possibilidade de comunicação de voz, dados e fax através de um único sistema;
- Possibilidade de uso de algoritmos de compactação que permitem que seja utilizada uma menor largura de banda por canal;
- Utilização de códigos de correção que melhoram a qualidade da transmissão, e
- Capacidade de criptografia dos dados que proporcionam aos usuários maior segurança das informações transmitidas.

2.4 Principais Tecnologias em Uso

A telefonia celular digital concentra atualmente um grande esforço tecnológico e empresarial, com diversos competidores tentando obter a hegemonia do mercado.

Por isso, distintos padrões de comunicação sem fio têm surgido, cada qual com suas particularidades buscando obter maiores performances e recursos [TAN 97].

Serão apresentadas as tecnologias GSM, CDMA, TDMA e CDPD que hoje correspondem aos principais padrões em uso.

2.4.1 GSM – *Global System for Mobile Communications*

Uma das desvantagens dos sistemas analógicos é sua incapacidade de atender demandas crescentes a custos baixos, oportunizando o surgimento da tecnologia digital. A tecnologia digital apresenta as vantagens de facilidade de sinalização, menores níveis de interferência e maior capacidade de atendimento de demanda.

Vários sistemas digitais começaram a ser desenvolvidos sem se preocuparem com padronização. Isso resultou em diversos problemas relacionados à compatibilidade, especialmente no que tange à tecnologia de rádio digital. O sistema GSM foi então projetado para tentar padronizar essa tecnologia.

Foi projetado sem nenhum compromisso de retrocompatibilidade com os sistemas analógicos, e operava inicialmente na freqüência de 900 MHz sendo mais tarde portado para a freqüência de 1800 MHz.

O padrão GSM define as funções e os requisitos das interfaces em detalhes, porém não se dedica ao *hardware.* Com isso busca limitar o mínimo possível os projetistas e também permitir que diferentes fornecedores participem do mercado.

Os sistemas digitais GSM têm ampla utilização no mercado europeu.

2.4.2 CDMA – *Code Division Multiple Access*

A tecnologia CDMA (*Code Division Multiple Access*) é baseada na divisão do espectro de freqüências que permite o uso simultâneo de múltiplas freqüências. Cada pacote de sinal é identificado com uma chave exclusiva, sendo que o receptor dá atenção apenas aos pacotes que possuem a chave destinada a ele, podendo capturá-los e demodular o sinal associado.

O sistema CDMA é bastante diferente dos demais e evita problemas como sincronização de tempo e alocação de canais. Além disso, é completamente descentralizado e totalmente dinâmico. Acabou sendo utilizado bastante por empresas. Antigamente, era um sistema utilizado por militares.

2.4.3 TDMA – *Time Division Multiple Access*

O padrão TDMA (*Time Division Multiple Access*) é uma tecnologia de transmissão digital que permite que um número de usuários acessem um único canal de rádio-freqüência sem interferirem uns nos outros, através da divisão de fatias de tempo exclusivas para cada usuário.

Atualmente o padrão TDMA para celulares divide cada canal em seis fatias de tempo, podendo multiplexar até três sinais distintos. Cada sinal utiliza duas fatias, o que permite o triplo da capacidade em relação ao AMPS.

Ele foi projetado para ser utilizado em diversos ambientes e situações, prevendo desde o uso com pouca mobilidade até o uso com deslocamentos em altas velocidades, como em uma rodovia expressa.

Também oferece uma variedade de serviços ao usuário, tais como comunicação por voz, de dados, fax, serviços de mensagens curtas (SMS) e mensagens do tipo *broadcast*.

2.4.4 CDPD – *Cellular Digital Packet Data*

O serviço digital abordado anteriormente, GSM, utiliza comutação por circuitos. Já o CDPD (*Cellular Digital Packet Data*) faz uso da comutação por pacotes e é diferente em relação às chamadas sem fio: no CDPD a cobrança é por *bytes* enviados e não por minuto de tempo de conexão como ocorre no sistema GSM.

Este sistema foi criado com base no AMPS com o qual é totalmente compatível.

2.5 Padrões Emergentes

Como esta é uma área em franco desenvolvimento e de grande interesse econômico, diversas tecnologias e protocolos têm surgido a uma velocidade muito grande.

Entre as principais tecnologias emergentes, destacam-se o i-Mode [ANY 00] e CDMA 2000 [TEL 00].

2.6 Serviços de Comunicação Pessoal

A idéia dos serviços de comunicação pessoal é permitir que o usuário possa comunicar-se de qualquer lugar com a utilização de um único número para identificação. Isto não ocorre na tecnologia AMPS, onde tem uma identificação para o dispositivo móvel e outra para o telefone fixo convencional.

Esse sistema, que se encontra em desenvolvimento, é denominado nos EUA como serviço PCS (*Personal Communications Service*) e nos demais paises como PCN (*Personal Communications Network*). Os serviços PCS utilizarão tecnologia celular, mas as células serão reduzidas (serão chamadas de microcélulas) permitindo que os telefones tenham o tamanho reduzido, se tornem mais leves e gastem menos energia para o funcionamento.

3 Serviço de Mensagens Curtas (SMS)

Short Message Service (SMS) é um serviço sem fio que possibilita a transmissão de mensagens alfanuméricas entre usuários de telefones móveis e sistemas externos, como correio eletrônico, *paging* e sistemas de correio de voz [IEC 00a]. O serviço SMS foi criado como parte do padrão GSM fase 1, e acredita-se que a primeira mensagem foi enviada em dezembro de 1992 [GSM 00]. A mensagem, que pode ser formada de palavras, números ou uma combinação alfanumérica, possui limitação de comprimento de 160 caracteres quando são utilizados alfabetos latinos e de 70 caracteres quando são usados alfabetos não latinos, como Chinês e Árabe. O SMS provê um mecanismo para transmitir mensagens curtas para e de terminais sem fio. O serviço faz uso de uma *Short Message Service Center* (SMSC) que atua como elemento de repasse e armazenamento das mensagens. A rede sem fios realiza o transporte das mensagens entre a central SMSC e os terminais sem fio.

Um diferencial entre o SMS e outros serviços de transmissão de mensagens de texto, como *paging*, é que os elementos que compõem o SMS são desenvolvidos com o objetivo de garantir a entrega das mensagens até o destino. Também, outra característica importante, é que o dispositivo móvel está apto a receber e enviar mensagens a qualquer hora, independente de estar ou não com uma chamada de voz ou uma transmissão de dados em progresso, e, quando são identificadas falhas temporárias, a mensagem fica armazenada na rede até que o destino volte a estar disponível.

Em relação à economia de recursos de rede, o serviço SMS é caracterizado pelo envio de pacotes sem consumo de banda e pelas transferências de mensagens de pequena largura de banda, que são dois fatores que tem relativa importância devido ao grande número de aplicações possíveis.

Inicialmente, as aplicações para o SMS estavam focadas no objetivo da eliminação dos *pagers* alfanuméricos, por permitir o envio de mensagens de propósito geral nos dois sentidos[1] e por estarem disponíveis alguns serviços de notificação. Com o amadurecimento da tecnologia e das redes, novos serviços como integração com correio eletrônico, fax e *paging* e serviços de informação, como boletins de previsão do tempo, surgiram e ampliaram o número de aplicações disponíveis aos usuários de terminais móveis.

---

[1] Embora hoje em dia já estarem disponíveis uma nova geração de dispositivos *pagers* que utilizando a tecnologia ReFlex permitem transmissão de mensagens nos dois sentidos.

Os dados dessas aplicações incluem o *download* de cartões SIM (*Subscriber Identify Module*) para fins de ativação, débito e alterações de perfil.

3.1 Benefícios Decorrentes do Uso de SMS

São vários os benefícios oferecidos pelo SMS, que pode, por exemplo, ser um fator de competitividade importante quando se trata de diferenciação de serviços oferecidos pelo provedor. Os benefícios do SMS para o provedor de serviços são os seguintes:

- Aumento do número de chamadas completadas em redes com e sem fio pela influência das capacidades de notificação do SMS;
- Uma alternativa a serviços de *paging* alfanuméricos;
- Possibilita que usuários corporativos tenham acesso a dados através de terminais móveis;
- Provisão de serviços de valor agregado como integração de *e-mail*, correio de voz, serviço de lembretes, agendamento de passagens aéreas e serviços de informação sobre taxas de câmbio e cotações de bolsas de valores;
- Provisão de serviços administrativos, como avisos de alteração de preço, entre outros.

Os benefícios da tecnologia SMS para os usuários incluem flexibilidade, facilidade de integração entre os serviços de mensagens e acesso a dados. Além disso, pode-se utilizar o terminal móvel dotado do SMS como uma extensão do computador.

3.2 Arquitetura e Elementos de Rede

A estrutura de rede básica do serviço SMS pode ser descrita como mostrado na fig. 3.1.

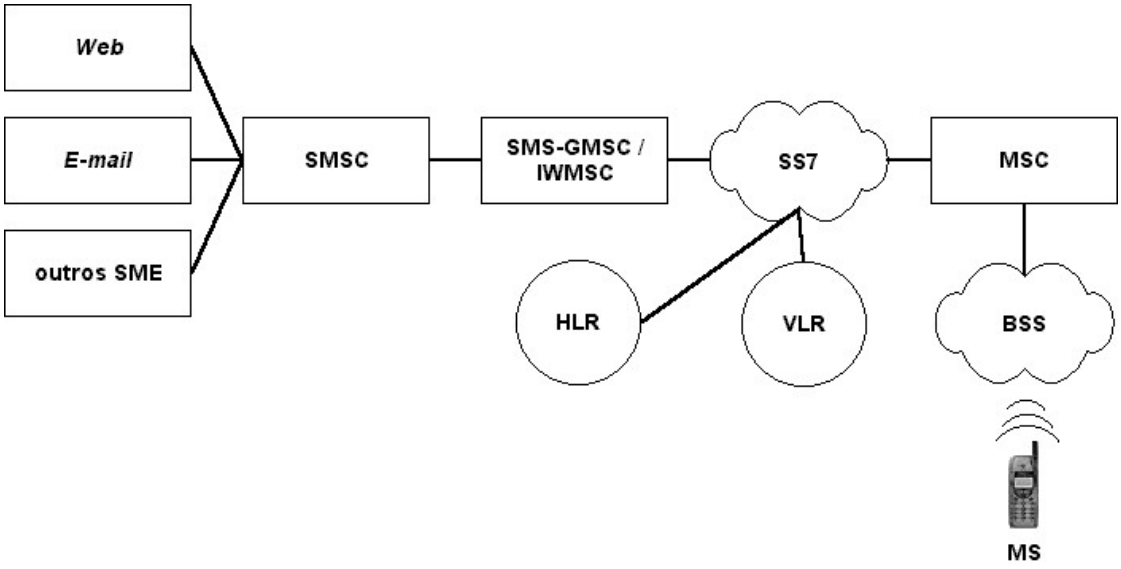

Figura 3.1 - Elementos de rede e arquitetura

Os componentes da estrutura de rede básica do serviço SMS serão descritos a fim de demonstrar suas funções.

SME – *Short Messaging Entities*

A entidade SME pode receber ou enviar mensagens curtas, podendo estar localizada em uma rede fixa, uma estação móvel ou outro centro de serviços.

SMSC – *Short Message Service Center*

A central SMSC é responsável pela retransmissão e armazenamento temporário de uma mensagem curta entre uma entidade SME e uma estação móvel.

HLR – *Home Location Register*

O registro HLR é um banco de dados usado para armazenamento permanente e gerenciamento de assinaturas e serviços de perfil. Quando interrogado pela central SMSC, o HLR provê a informação de roteamento para o assinante indicado. O registro HLR também informa a central SMSC, que não teve sucesso anteriormente ao tentar despachar a mensagem, e que a estação móvel agora está acessível.

MSC – *Mobile Switching Center*

A central de chaveamento MSC realiza as funções de chaveamento do sistema e controla chamadas para e de outro telefone e sistema de dados.

SMS-GMSC – SMS-*Gateway Mobile Switching Center*

Um *gateway* SMS-GMSC é capaz de receber uma mensagem de uma central SMS, interrogar um banco de dados HLR para obter informações de roteamento, e despachar a mensagem para a central de chaveamento (MSC) onde se encontra a estação móvel receptora.

SMS-IWMSC - SMS-*Interworking Mobile Switching Center*

A central de interconexão SMS-IWMSC é capaz de receber uma mensagem da rede de dispositivos móveis e enviá-la para a central SMS apropriada.

VLR – *Visitor Location Register*

O *Visitor Location Register* (VLR) é um banco de dados que contém informações temporárias sobre assinantes. Esta informação é necessária à central de chaveamento (MSC) para o serviço a assinantes visitantes.

BSS – *Base Station System*

Todas funções de transmissão e recepção via rádio são realizadas no *Base Station System* (BSS). O sistema BSS consiste de *Base-Station Controllers* (BSCs) e as *Base-Transceiver Stations* (BTSs), sendo o principal responsável em transmitir tráfego de voz e dados entre as estações móveis.

MS – *Mobile Station*

Uma estação móvel (*Mobile Station)* é um terminal sem fio capaz de receber e originar mensagens curtas bem como chamadas de voz.

SS7 – *Signaling System 7*

O sistema SS7 fornece a informação necessária para estabelecer e manter chamadas telefônicas envolvendo redes separadas, e não apenas na própria rede do telefone originador.

3.3 Elementos de Sinalização

A camada MAP (*Mobile Application Part*) define as operações necessárias para suportar o serviço SMS. Ambos padrões americanos e internacionais possuem uma camada MAP definida utilizando os serviços do conjunto de facilidades de controle de

transações SS7 (*Signaling System 7*) [IEC 00b]. O padrão americano é definido pela *Telecommunication Industry Association* e é referenciada como IS-41. Já o padrão internacional é definido pela *European Telecommunication Standarts Institute* e é referenciado como GSM MAP.

As seguintes operações básicas MAP são necessárias para prover um serviço de mensagens curtas cliente a cliente:

- Requisição de informação de roteamento – antes de tentar realizar o despacho da mensagem, a central SMSC necessita buscar a informação de rota para determinar a central de chaveamento (MSC) que está servindo a estação móvel no momento da tentativa de entrega;
- Entrega de mensagem ponto a ponto (*point-to-point*) – o mecanismo provê uma forma de a central SMSC transferir a mensagem à central de chaveamento MSC que serve a estação móvel endereçada. A central de chaveamento MSC também é responsável por entregar uma mensagem para uma estação móvel sempre que esta está registrada, ou quando a estação é contactada em uma chamada de voz ou dados. A operação de entrega da mensagem curta gera um serviço de entregas confirmado. A operação trabalha em tandem com o subsistema estação base enquanto a mensagem está sendo repassada da central de chaveamento MSC a estação móvel. Por isso, o resultado da operação compreende ou sucesso (entrega ao dispositivo móvel) ou falha causada por uma das várias razões possíveis. A entrega da mensagem ponto a ponto é realizada por via do uso dos mecanismos SMD-PP (*Short Message Delivery Point-to-Point*) e *ForwardShortMessage*, em IS-41 e GSM, respectivamente;
- Indicação de mensagem em espera – esta operação é ativada quando a entrega de uma mensagem falhou devido a um problema temporário, e permite que a central SMS solicite ao HLR que seja adicionado seu endereço de central SMS na lista de centrais SMS que devem ser informadas quando o dispositivo móvel tornar-se acessível novamente;
- *Service Center Alert* – a operação provê uma forma de o registro HLR informar à central SMSC, que anteriormente inicializou sem sucesso uma tentativa de entrega da mensagem para especificar a estação móvel, e que a estação móvel é agora reconhecida pela rede móvel para ser

acessível. Este alerta é sinalizado pelo uso da notificação SMS e mecanismos do centro de serviços de alerta em IS-41 e GSM respectivamente.

3.4 Elementos de Serviço

A tecnologia SMS compreende alguns elementos de serviço relevantes para a recepção e submissão de mensagens curtas.

- Período de validade – o período de validade indica quanto tempo a central SMSC deve garantir o armazenamento da mensagem curta antes de distribuir para o destinatário;
- Prioridade – prioridade é o elemento de informação provido pela entidade SME para indicar a prioridade da mensagem.

Além disso, SMS provê um registro da hora reportando o tempo da submissão da mensagem e uma indicação para o receptor se há mais mensagens para enviar (GSM) ou o número de mensagens adicionais para enviar (IS-41).

3.5 Serviços de Assinante

O serviço SMS compreende dois serviços básicos ponto a ponto:

- Mensagem curta originada pelo dispositivo móvel (MO-SM);
- Mensagem curta destinada ao dispositivo móvel (MT-SM).

As mensagens curtas originadas pelo dispositivo móvel (MO-SM) são transportadas do *handset* para a central SMSC e pode ser destinada a outro assinante de dispositivo móvel ou para assinantes em uma rede fixa como redes de *pager* ou redes de *e-mail*. As mensagens curtas destinadas ao dispositivo móvel (MT-SM) são transportadas da central SMSC para o receptor e podem ser submetidas para a SMSC por outros assinantes móveis via MO-SM ou outras fontes como sistemas de *voice-mail*, redes de *pager*, ou operadoras.

Para cada mensagem curta destinada ao dispositivo móvel (MT-SM), um relatório é sempre retornado para a central SMSC confirmando a entrega da mensagem ao receptor ou informando a SMSC que ocorreu falha, identificando o motivo. Similarmente, para cada mensagem curta originada pelo dispositivo móvel (MO-SM), um relatório é sempre

retornado para o originador também confirmando a entrega da mensagem a central SMSC ou informando a ocorrência de falha e o motivo pelo qual ocorreu.

Dependendo do método de acesso e da codificação de dados da portadora, o SMS ponto a ponto admite até 190 caracteres para uma entidade SME. Para mensagens que requerem entrega imediata, somente uma tentativa de entrega é realizada. Para mensagens que não requerem uma entrega imediata, uma ou mais tentativas de entrega são feitas até um reconhecimento ser recebido.

O serviço SMS pode ser usado como meio de transporte de uma série de outros serviços de mensagens, além de poder proporcionar aos dispositivos móveis um acesso seguro e envio de informações pela Internet ou Intranet de forma rápida e com custos atraentes.

A infraestrutura genérica da rede para a implementação dos serviços SMS é demonstrada abaixo (fig. 3.2):

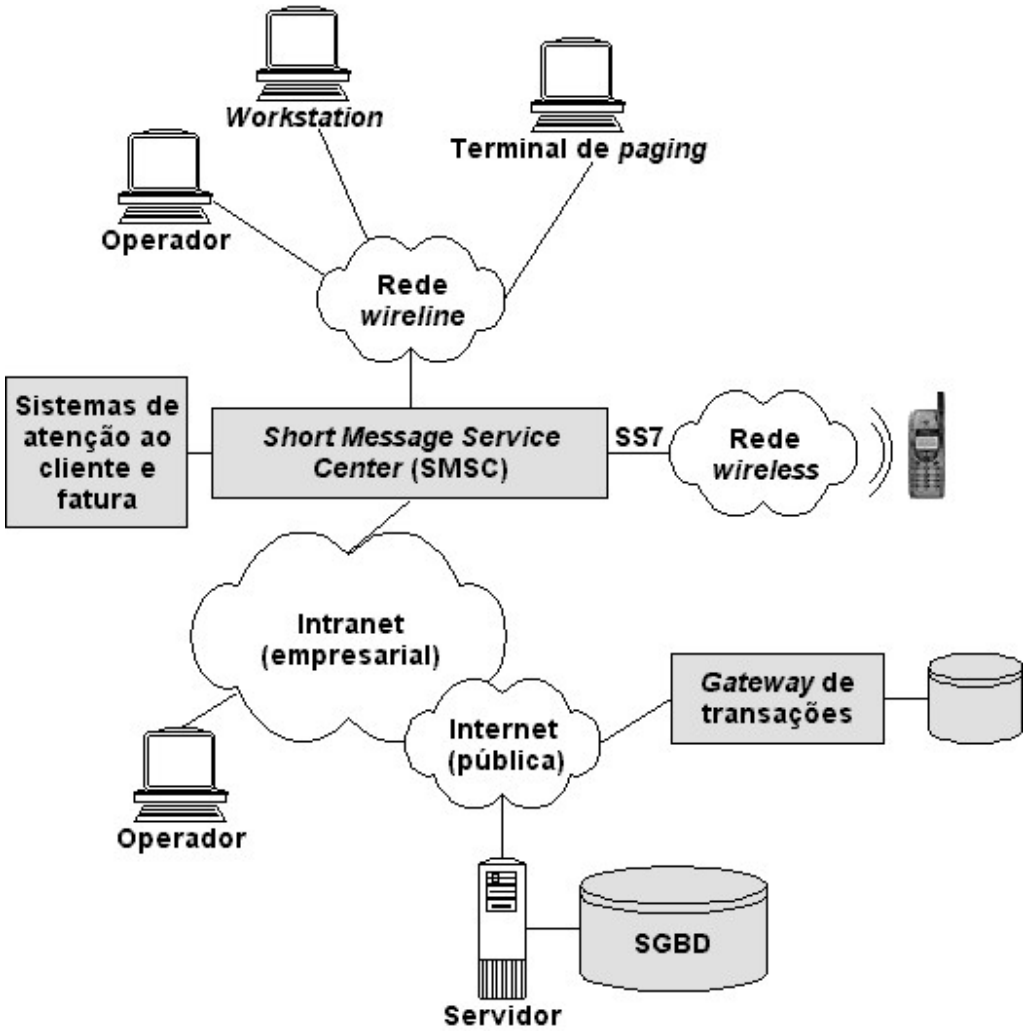

Figura 3.2 - Infraestrutura genérica de uma rede SMS

Algumas das aplicações potenciais da tecnologia SMS, utilizando ambos MT-SM e MO-SM são descritas brevemente a seguir:

- Serviços de notificação – estes serviços correspondem à maior parte dos serviços SMS desenvolvidos. Exemplos de serviços de notificação usando SMS: *voice/*fax *message notification* que indica que mensagens de *e-mail* de voz estão presentes em uma determinada caixa de correio de voz, notificação de *e-mail* que indica que mensagens de *e-mail* estão presentes em uma caixa de *e-mail*, e serviços de lembrança de compromissos que permitem lembrar por encontros e compromissos agendados;
- Serviços de *e-mail* – serviços de *e-mail* existentes (ex. SMTP, X.400) podem estar facilmente integrados com serviços SMS;
- Serviços de *paging* – serviços de *paging* (ex. TAP, TNPP, TDP) integrados com SMS permitem a assinantes de serviços digitais sem fio serem acessados por outras interfaces de *paging* existentes;
- Serviços de informação – uma larga variedade de serviços de informação podem ser providos pelo SMS, incluindo previsões do tempo, informações sobre tráfego, informações sobre entretenimento (cinema, teatro, concertos), informações financeiras (cotações de bolsas de valores, taxas de câmbio, *home banking*) e serviços de listas.

3.6 Serviços de Dados Móveis

A central SMSC também pode ser utilizada para prover pequena quantidade de dados. Os dados sem fio podem estar em serviços interativos onde chamadas de voz estão envolvidas. Alguns exemplos de serviços desta natureza incluem controle frota de entrega, gerenciamento de inventários, confirmação de itinerários, processamento de ordens de vendas e gerenciamento de contato com clientes.

3.7 Gerenciamento e Atenção com Clientes

A central SMSC também pode ser utilizada para transferir dados binários que podem ser interpretados pela estação móvel sem apresentação ao cliente. Esta capacidade permite aos operadores administrar seus clientes, provendo capacidades para programar a

estação móvel. Exemplos destes serviços incluem programação do terminal móvel, que permite que características personalizadas de assinantes sejam transferidas para os terminais móveis, além do aviso de custos, que habilita o SMS ser utilizado para reportar custos envolvidos com a chamada telefônica (a ser pago na conta telefônica).

4 Protocolo de Aplicações Sem Fio (WAP)

O *Wireless Application Protocol* (WAP) é um ambiente de aplicação e um grupo de protocolos de comunicação para dispositivos sem fio, criado para permitir – de uma forma independente de fabricante, vendedor e tecnologia – acesso à Internet e a um conjunto de serviços de telefonia avançados [IEC 00]. O protocolo WAP pode ser visto como sendo a ponte que interliga o mundo dos dispositivos móveis e a Internet – assim como as Intranets das corporações – e oferece a habilidade para entregar uma ilimitada gama de serviços móveis de valor agregado para os usuários, independente da rede e do terminal móvel. Assim, o usuário pode acessar a mesma riqueza de informação disponível em um microcomputador de mesa através de um dispositivo de bolso, com a vantagem de que o conteúdo está disponível ao mesmo tempo em que o usuário se movimenta.

4.1 Origem do Protocolo

A especificação do WAP teve origem em 1997, quando um grupo de empresas formado pela Ericsson, Nokia, Motorola e Unwired Planet deram origem ao chamado WAP *Forum*[2], que ainda hoje é o órgão responsável pela publicação das especificações assim como pela homologação de produtos WAP. A proposta inicial do WAP *Forum* foi a de definir uma ampla especificação para a indústria a fim de permitir o desenvolvimento de aplicações sobre redes de comunicação sem fio. O WAP *Forum* é um fórum aberto a qualquer participante da indústria relacionada à tecnologia WAP – como fabricantes de terminais móveis, infra-estrutura, provedores de serviços e de conteúdo, operadoras e desenvolvedores de *software* – e atualmente possui mais de 550 membros, incluindo Microsoft, America Online, Intel e IBM.

As diretrizes definidas pelo WAP *Forum* têm a finalidade de alcançar os seguintes objetivos [RIB 00]:

- Tornar disponível conteúdo da Internet e serviços de dados avançados aos telefones celulares digitais ou outros terminais sem fio;
- Criar uma especificação global de protocolos que trabalhe através de diferentes tecnologias de redes sem fio;
- Incentivar a criação de conteúdo e aplicações que são providos através de uma variedade de redes sem fio e terminais móveis;

[2] Sítio oficial do WAP *Forum* na *Web*: http://www.wapforum.org/.

- Adotar, sempre que possível, e estender padrões e tecnologias existentes.

O WAP *Forum* possui um programa de certificação de produtos onde fabricantes se credenciam com o objetivo de obter um certificado de conformação de seus produtos com a especificação WAP corrente (atualmente WAP versão 1.2). Da mesma forma possui um programa de registro de desenvolvedor e também um programa de verificação de conteúdo [WAP 00a].

O telefone celular digital, o *Personal Digital Assistant* (PDA), o *pager*, além de outros dispositivos sem fio homologados pelo WAP *Forum*, formam um grande conjunto de aparelhos que se beneficiam de serviços oferecidos pela especificação WAP, como *home banking*, comércio eletrônico, correio eletrônico, notícias e entretenimento.

4.2 Benefícios do Uso de WAP

Para as operadoras de redes sem fio, WAP promete reduzir custos e incrementar a base de usuários tanto pela melhoria de serviços existentes quanto pela criação de uma nova gama de serviços e aplicações agregadas. Novas aplicações podem ser introduzidas de forma rápida e fácil sem a necessidade adicional de infra-estrutura ou modificação no terminal móvel. Isto possibilita às operadoras diferenciarem-se pela forma como personalizam seus serviços e informações, fazendo disto uma interessante vantagem competitiva [IEC 00].

Os usuários de terminais móveis se beneficiam pela facilidade e segurança no acesso a informações relevantes disponíveis na Internet, sendo possível, por exemplo, o acesso a informações contidas em bancos de dados de corporações para uso em aplicações de *m-commerce*[3]. Outro benefício se refere à grande variedade de fabricantes que já suportam a iniciativa do WAP *Forum*, permitindo que usuários tenham significante liberdade na escolha de terminais móveis e aplicações.

Além disso, as inúmeras aplicações propiciadas pelo WAP-*Push* (v. seção 4.9), como recebimento de informações meteorológicas e de condições de rodovias, trazem uma vantagem adicional às tradicionais aplicações da WWW, permitindo que o usuário receba informações relevantes em tempo real sem a necessidade de requisitá-las [WAP 99].

---

[3] Comércio móvel (*m-commerce*) – possibilidade de consultar preços e efetuar compras, entre outras operações comerciais, a partir de um dispositivo móvel.

O WAP utiliza padrões da Internet como XML (*Extensible Markup Language*), UDP (*User Datagram Protocol*) e IP (*Internet Protocol*). Muitos destes protocolos são baseados em padrões da Internet como HTTP (*Hypertext Transfer Protocol*) e TLS (*Transport Layer Security*), mas têm sido otimizados para as restrições encontradas em ambientes sem fio: pequena largura de banda, alta latência e baixa estabilidade de conexão [BRA 98a, BRA 98, POS 80, FIE 99].

Padrões utilizados na Internet, como HTML, HTTP, TLS e TCP (*Transmission Control Protocol*) são ineficientes sobre redes de terminais móveis, pois requerem uma grande quantidade de dados em formato texto a serem enviados [IEC 00]. Dessa forma, o conteúdo de um arquivo HTML padrão não pode ser mostrado de forma eficiente em telas de tamanho reduzido, como as de telefones móveis e *pagers*. Como decorrência, uma série de novos padrões e protocolos foram desenvolvidos, tendo por base os já consagrados na Internet e *Web*. A tabela abaixo relaciona os principais protocolos e padrões WAP, associando-os por funcionalidade, aos correspondentes na *Web*. A partir da seção 4.6 serão detalhados tais protocolos.

Tabela 4.1 - Comparação entre *Web* e WAP

Web	**WAP**
HTML Java*Script*	WAE – WML WML*Script*
HTTP – Sessão Transação	WSP – Sessão WTP – Transação
TLS – Segurança	WTLS – Segurança
UDP – Datagrama	WDP – Datagrama
IP – Endereço	*Bearer* – Endereço

No padrão WAP, além da utilização de transmissão binária, permitindo que os dados sejam comprimidos, as sessões têm capacidade de adequação a problemas de cobertura intermitente e podem operar sobre uma ampla variedade de formas de transporte.

Também, fazendo-se uma comparação com relação aos recursos disponíveis em um microcomputador pessoal, percebe-se que além das restrições citadas anteriormente, existem outras dificuldades a serem superadas, como a menor capacidade de processamento das CPUs, quantidade reduzida de memória RAM e/ou ROM, consumo de potência restrito, visores de menor tamanho e dispositivos de entrada diferentes, o que justifica a necessidade

de otimização e adaptação de padrões amplamente utilizados na Internet, como os citados acima [RIB 00].

As linguagens WML (*Wireless Markup Language*) e WML*Script* (*Wireless Markup Language Script*) atualmente são utilizadas para a produção de conteúdo WAP, podendo este ser visualizado tanto em telas com apenas duas linhas, disponíveis em um dispositivo básico, como também em telas totalmente gráficas e maiores que acompanham os dispositivos mais recentemente lançados.

O conjunto de protocolos WAP é desenvolvido para minimizar o tamanho de largura de banda requerido e maximizar o número de tipos de redes sem fio que podem distribuir conteúdo WAP [IEC 00]. Devido a ser projetado para otimizar o trabalho através de uma variedade de interfaces móveis, o WAP permite que um número maior de provedores de serviço e fabricantes de dispositivos móveis trabalhe em uma única especificação, e, além disso, tem-se a certeza de que cada aplicação, desenvolvida utilizando-se um padrão, trabalhará em uma diversidade de tipos de redes. Os fabricantes podem utilizar o mesmo *software* em todas as linhas de produto, o que reduz o tempo de desenvolvimento e simplifica o suporte. Também, com uma interface móvel independente, é possível, de forma rápida e fácil, estender a especificação.

Além dessas características, a especificação do protocolo WAP é projetada com o objetivo de encorajar a fácil e aberta interoperabilidade entre seus componentes fundamentais [RIB 00]. Assim, qualquer componente que possui compatibilidade com a especificação WAP pode interoperar com qualquer outro componente também compatível com o WAP. Provedores de conteúdo e serviços podem escolher equipamentos e *software* de múltiplos fabricantes, selecionando cada parte da solução de acordo com as suas necessidades.

4.3 O Modelo de Rede WAP

O modelo de programação WAP (fig. 4.1) é similar ao da *World-Wide Web* [WAP 98].

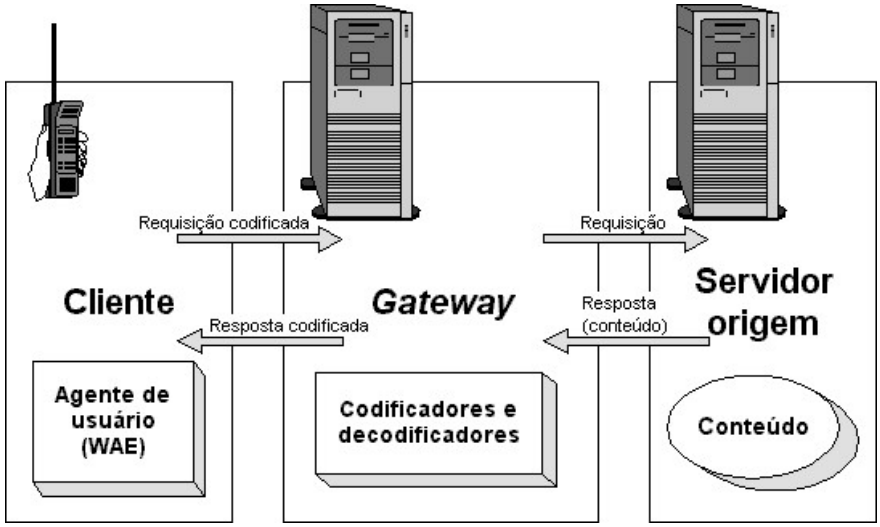

Figura 4.1 – Modelo de programação WAP

O conteúdo e as aplicações WAP são especificadas em um conjunto de formatos padronizados e baseados nos formatos já familiares da WWW, sendo transportados utilizando um conjunto de protocolos de comunicação também baseados nos utilizados na *Web*. Um *microbrowser* disponível no terminal sem fio, semelhante a um *Web browser*, coordena a interface com o usuário.

O protocolo WAP define um conjunto de componentes que possibilitam a comunicação entre os terminais móveis e os servidores de rede, o que inclui:

- Um modelo de nomes padrão [WAP 98a] – URI (*Universal Resource Identifier*) e URL (*Uniform Resource Locator*), que consistem em padrões utilizados na WWW, são utilizados. O recurso URL é usado para identificar o conteúdo WAP nos servidores de origem e URI identifica recursos locais em um dispositivo, como por exemplo, as funções de controle de chamadas;
- Topologia de conteúdo – Todo conteúdo WAP é caracterizado por um tipo específico consistente com a topologia WWW. Isto permite que agentes de usuário WAP processem de forma correta o conteúdo, pois o mesmo é baseado em um tipo pré-definido;
- Padrão de formatos de conteúdo – Os formatos de conteúdo WAP são baseados na tecnologia WWW, incluindo informação de calendário, cartão de negócios eletrônico, imagens e linguagem de *script*;

- Padrão de protocolos de comunicação – Os protocolos de comunicação WAP permitem a comunicação entre as requisições feitas pelo *browser* do terminal móvel e o servidor de rede *Web*.

O protocolo WAP utiliza a tecnologia *proxy*[4] para realizar a conexão entre o domínio sem fio e a WWW.

Na especificação do WAP, o acesso a uma determinada aplicação é conseguido através do envio de uma requisição do dispositivo móvel (na forma de uma URL) ao *proxy* WAP. Isto permite que o conteúdo e as aplicações sejam armazenados em servidores WWW comuns e que este conteúdo e aplicação sejam desenvolvidos utilizando tecnologias também usadas na WWW, como CGI *Scripting*.

O *proxy* WAP (fig. 4.2), que é a interface entre a rede da operadora e a Internet, deve incluir as seguintes funcionalidades [RIB 00]:

- Um mecanismo, normalmente denominado *gateway* de protocolos, cuja função é traduzir as requisições provenientes da pilha de protocolos WAP para o conjunto de protocolos WWW (HTTP e TCP/IP);
- Codificadores e decodificadores de conteúdo, que têm o objetivo, respectivamente, de codificar e decodificar a informação desenvolvida em WML, para que esta seja transportada através da rede sem fio de forma compactada, o que diminui o tráfego de dados necessário.

Um exemplo de modelo de rede WAP [WAP 98] pode ser visualizado através da fig. 4.2.

[4] Agente intermediário que se encarrega de redirecionar a requisição de serviços ao verdadeiro servidor, atuando em nome deste e dessa forma, facilitando a requisição ao cliente.

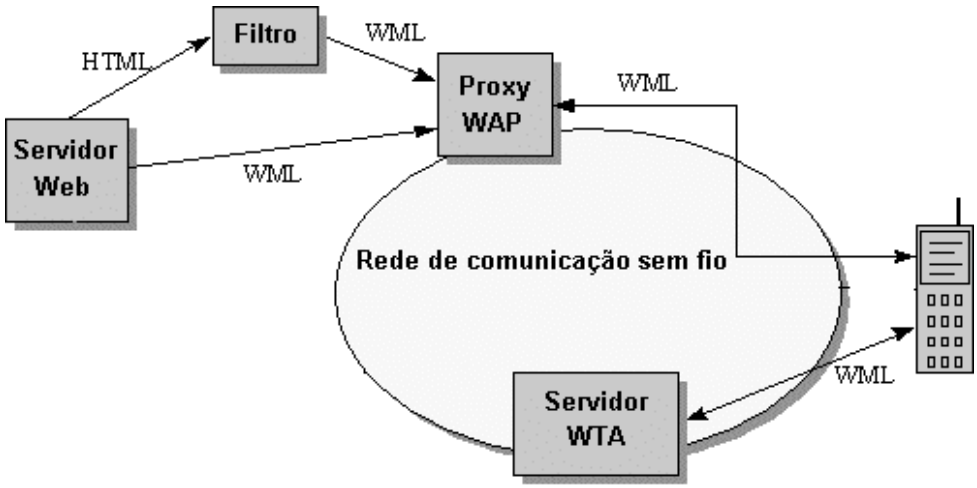

Figura 4.2 - Exemplo de modelo de rede WAP

Neste exemplo, o cliente WAP se comunica com dois servidores na rede sem fio. O *proxy* WAP traduz as requisições WAP para requisições WWW e, desse modo, permite que o cliente WAP submeta requisições ao servidor *Web*. O *proxy* também codifica as respostas do servidor *Web* em um formato binário compactado entendido pelo cliente. Podemos também observar que há necessidade de comunicação entre o cliente e o *proxy* WAP sempre que o usuário deseja acessar serviços disponibilizados na Internet. Já o acesso ao servidor WTA (*Wireless Telephony Application*), que provê serviços de telefonia, é feito diretamente. Além disso, percebe-se que existem duas formas de acesso ao servidor WWW. Se este servidor apresentar o conteúdo no formato WML, as informações são enviadas diretamente ao *proxy* WAP; caso contrário, quando as informações estão disponíveis em HTML, é necessário que este conteúdo seja filtrado antes. A função deste filtro, conforme pode ser observado na figura, é a de converter conteúdo no formato HTML em WML e posteriormente enviá-lo ao *proxy* WAP [RIB 00].

O uso do *gateway* de protocolos (ou *gateway* WAP) permite que o conteúdo e as aplicações, criadas para serem acessadas através de um dispositivo móvel, possam ser hospedadas em um servidor WWW comum, beneficiando-se de toda a estrutura e tecnologia da Internet existente.

A comunicação entre o servidor WWW e o *gateway* WAP é realizada através do protocolo HTTP v1.1. Dessa forma, os usuários podem utilizar URLs diferentes para acessar conteúdo HTML ou WML; ou até mesmo usar um único URL para disponibilizar ambos formatos, deixando a cargo do servidor WWW a escolha de qual conteúdo enviar (decisão tomada de acordo com o navegador utilizado pelo cliente) [SEC 00].

4.4 Modelo de Segurança

O WAP disponibiliza uma infraestrutura de segurança flexível cujo foco é prover uma conexão segura entre o cliente WAP e o servidor, podendo prover uma segurança do início ao fim durante todo o trajeto entre dispositivos com protocolo WAP. Se o *browser* e o servidor onde reside o conteúdo desejarem segurança durante todo o percurso, é necessário que haja uma comunicação direta entre eles utilizando protocolos WAP. Segurança do início ao fim também pode ser realizada se o *proxy* WAP é confiável, por exemplo, se está localizado no mesmo local fisicamente seguro com o servidor de conteúdo.

4.5 Arquitetura de Protocolos WAP

A arquitetura WAP provê um ambiente escalável e extensível para o desenvolvimento de aplicações para dispositivos de comunicação móveis. Isto é realizado através da disposição da pilha de protocolos em camadas (fig. 4.3). Cada uma das camadas da arquitetura é acessível e provê serviços às camadas acima, bem como a outros serviços e aplicações.

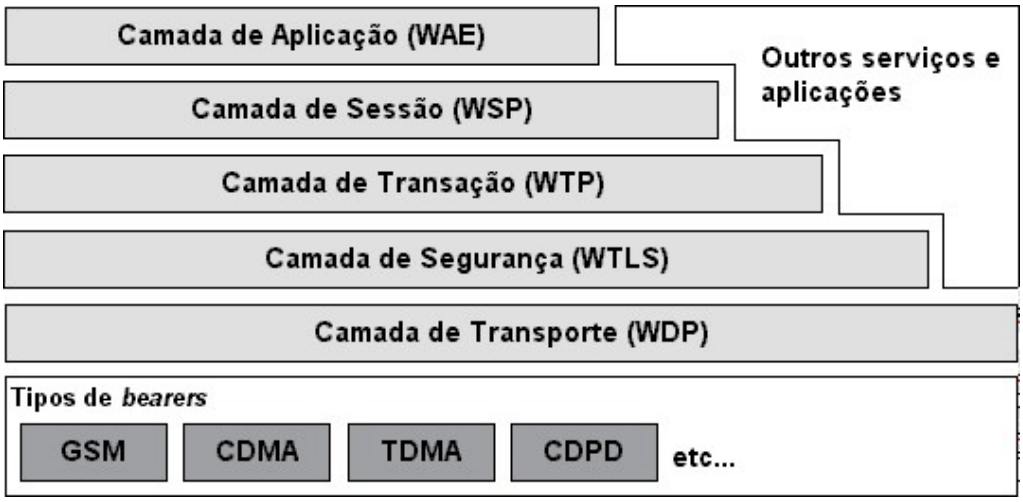

Figura 4.3 - Pilha de protocolos WAP

Isto permite que aplicações compartilhem serviços através de um conjunto de interfaces bem definidas, e que aplicações externas, como correio eletrônico e agendas, possam acessar qualquer uma das camadas diretamente. A seguir é realizada uma breve descrição dos vários elementos que compõem a pilha de protocolos WAP.

4.5.1 Camada de Aplicação – WAE

A camada *Wireless Application Environment* (WAE), consiste em um ambiente de aplicação de propósito geral baseado na combinação de duas tecnologias em crescente evidência: WWW e telefonia móvel. O principal objetivo da camada WAE é o de estabelecer um ambiente que permita a desenvolvedores de *software*, operadoras e provedores de conteúdo, construírem aplicações e serviços que possam ser disponibilizados em uma variedade de plataformas sem fio de maneira eficiente [RIB 00].

A camada WAE inclui um navegador denominado *microbrowser*, que possui as seguintes funcionalidades:

- *Wireless Markup Language* (WML) – Uma linguagem de marcação semelhante a HTML, mas otimizada para o uso em terminais móveis;
- WML*Script* – Uma linguagem de *script* semelhante a Java*Script*;
- *Wireless Telephony Application* (WTA, WTAI) – serviços de telefonia e programação de interfaces;
- Formatos de conteúdo – Um conjunto de formatos de dados bem definidos, incluindo imagens, campos de agenda de telefones e informação de calendário.

O agente de usuário WML, que é qualquer *software* ou dispositivo que interpreta conteúdo WML, atualmente é visto como o componente fundamental da camada WAE.

O WTA especifica uma estrutura de aplicação para serviços de telefonia. Esta estrutura reside em um agente de usuário WTA, o qual estende o agente WML, adicionando capacidade de comunicação com serviços que interagem com componentes WTA, a saber [RIB 00]:

- Um componente de manipulação de eventos de rede para prover, por exemplo, a sinalização de uma chamada entrante;
- Um mecanismo de armazenamento persistente (repositório);
- Uma interface para funções locais relacionadas com telefonia, tais como estabelecimento de chamada ou lista de telefones.

A interface genérica para funções relacionadas à telefonia é especificada pela WTAI (*Wireless Telephony Application Interface*).

Uma descrição mais detalhada sobre a camada WAE pode ser obtida através do acesso a documentação disponível no sítio do WAP *Forum* [WAP 98a].

4.5.2 Camada de Sessão – WSP

O protocolo *Wireless Session Protocol* (WSP), implementa funções equivalentes às definidas na camada de sessão do modelo OSI (*Open Systems Interconnection*), provendo à camada WAE uma forma eficiente de troca de dados entre aplicações através de uma interface consistente para dois tipos de serviços de sessão. O primeiro é um serviço orientado a conexão que opera sobre o protocolo da camada de transação WTP. O segundo é um serviço sem conexão que transporta datagramas (WDP).

Os protocolos correntemente disponíveis para o WSP definem serviços de aplicações de navegação (WSPB – *Wireless Session Protocol Browsing*), provendo a seguinte funcionalidade:

- Funcionalidade e semântica do HTTP v1.1 em um modo de codificação compacto;
- Estado de sessão de vida longa (*Long-lived Session State*) – o "tempo de vida" de uma sessão é independente da duração de uma conexão de transporte ou da continuidade de operação de uma rede sem fio;
- Suspensão e religação com migração da sessão – situação de uso comum em dispositivos portáteis;
- Facilidades para o envio de dados sem demanda (*data push*) – *push* é um serviço de sessão capaz de "empurrar" informação ao cliente sem ter sido por ele solicitada e *pull* é um serviço de sessão em que o usuário "puxa" informação do servidor (assim como o HTTP v1.1, mecanismo tradicional de navegação na WWW);
- Negociação de parâmetros de protocolo.

Os protocolos da família WSP são otimizados para redes de portadores de pequena largura de banda e com latência relativamente longa. O protocolo WSPB (ou WSP) é desenvolvido para permitir que um *proxy* WAP conecte um cliente WSPB (ou WSP) a um servidor HTTP padrão [WAP 98b].

4.5.3 Camada de Transporte – WTP, WTLS e WDP

O conjunto das camadas WTP, WTLS e WDP possui correspondência de funcionalidade em relação à camada de transporte do modelo OSI.

A camada de transporte do modelo WAP pode ser implementada de diversas formas, segundo o tipo de serviço desejado pela camada de sessão (WSP). As camadas WTP e WTLS poderão ser excluídas da pilha de protocolos WAP caso a aplicação não necessite de suas funcionalidades. Também a camada WDP, que realiza o transporte de datagramas através das diferentes tecnologias de redes sem fio sobre serviços de rede não IP, pode ser substituída pela camada UDP sobre IP, como na Internet.

A camada WTP (*Wireless Transaction Protocol*) se situa acima da camada WDP, tendo como função tornar disponível um protocolo orientado a transação que é apropriado para implementação em estações móveis [WAP 98c]. O protocolo WTP opera de forma eficiente sobre redes de datagrama sem fio, seguras ou não, e apresenta as seguintes características:

- Três classes de serviços de transação:
  - Classe 0: pedido unidirecional não confiável;
  - Classe 1: pedido unidirecional confiável;
  - Classe 2: transação bidirecional confiável (serviço mais utilizado);
- Confiabilidade usuário-usuário (opcional) – confirmação em cada mensagem recebida;
- Dado fora de banda (opcional) – inserido dentro de pacotes de reconhecimento;
- Concatenação de PDU (*Protocol Data Unit*) e atraso proposital nos reconhecimentos a fim de reduzir o número de mensagens enviadas;
- Transações assíncronas.

Uma das aplicações que faz uso do serviço orientado à transação (classe 2) é o WWW que hoje é considerado uma das aplicações mais utilizadas na Internet.

O *Wireless Transport Layer Security* (WTLS), é um protocolo de segurança baseado em SSL (*Secure Sockets Layer*), atualmente denominado TLS, que provê serviço de transporte seguro e é requerido por várias aplicações, incluindo *m-commerce*. É utilizado em

conjunto com protocolos da camada de transporte da arquitetura WAP, e apresenta algumas características [RIB 00]:

- Integridade de dados – possui facilidades para garantir que dados enviados entre o terminal e o servidor não sejam corrompidos ou alterados;
- Privacidade – garante que o dado transmitido entre o terminal e o servidor seja privado, impossibilitando que terceiros o compreendam;
- Autenticação – contém facilidade para estabelecer a autenticação do terminal e servidor de aplicação;
- Proteção *Denial-of-Service* – realiza detecção e rejeição de dados duplicados e não corretamente verificados.

Se as camadas inferiores da rede já implementam uma ou algumas dessas facilidades, a camada WTLS pode ser excluída da pilha de protocolos WAP. O comércio eletrônico é um exemplo típico de aplicação que faz uso das funcionalidades da WTLS [WAP 98d].

A camada WDP, equivalente a camada UDP na arquitetura TCP/IP, opera sobre tipos variados de *bearer* (SMS, USSD, CSD, CDPD, IS-136, GPRS). Os protocolos desta camada oferecem um serviço de transporte de datagramas consistente para as camadas superiores, abstraindo destes protocolos o tipo de rede ao qual se apóia através de uma interface comum. Uma vez que os protocolos da camada WDP provêm uma interface comum para os protocolos de camadas superiores, as camadas de segurança, sessão e aplicação estão habilitadas para funcionar independentemente da subjacente rede sem fio. Isso permite adaptar a camada de transporte para características específicas do *bearer* subjacente. Por manter a interface da camada de transporte e as características básicas consistentes, a interoperabilidade global pode ser conseguida utilizando *gateways* intermediários [WAP 98e].

A especificação WAP ainda define o protocolo WCMP (*Wireless Control Message Protocol*), que possui semelhança ao ICMP (*Internet Control Message Protocol*) da pilha TCP/IP. O WCMP é utilizado para tratar erros do WDP como "destino não alcançado" e "codificação errada de parâmetros no cabeçalho do datagrama WDP".

4.6 *Bearers*

Os protocolos WAP são desenvolvidos para operar sobre uma variedade de serviços de *bearers*, incluído SMS, circuitos virtuais comutados e datagramas. Os *bearers* oferecem diferentes níveis de qualidade de serviço com relação a vazão, taxas de erro e atrasos, sendo os protocolos WAP desenvolvidos para compensar ou tolerar estes níveis de variação do serviço.

Uma vez que a camada WDP provê a convergência entre os serviços dos *bearers* e o resto da pilha WAP, a especificação da WDP lista os *bearers* que são suportadas e as técnicas utilizadas para permitir que protocolos WAP rodem sobre cada uma delas.

4.7 Outros Serviços e Aplicações

A arquitetura em camadas WAP possibilita que outros serviços e aplicações utilizem as características da pilha WAP através de um conjunto de interfaces bem definidas. Aplicações externas podem acessar as camadas de sessão, transação, segurança e transporte diretamente. Isto permite que a pilha de protocolos WAP seja usada por aplicações e serviços não correntemente especificados pelo WAP, mas consideradas de grande valor pelo mercado de redes sem fio. Por exemplo, aplicações, como *e-mail*, calendário, agenda de telefones, bloco de notas, comércio eletrônico, ou serviços como páginas amarelas, podem ser desenvolvidos para utilizar os protocolos WAP.

4.8 Exemplos de Configuração da Tecnologia WAP

Espera-se que a tecnologia WAP seja útil em aplicações e serviços além dos previstos pelo WAP *Forum*. A fig. 4.4 descreve várias possíveis pilhas de protocolos utilizando a tecnologia WAP.

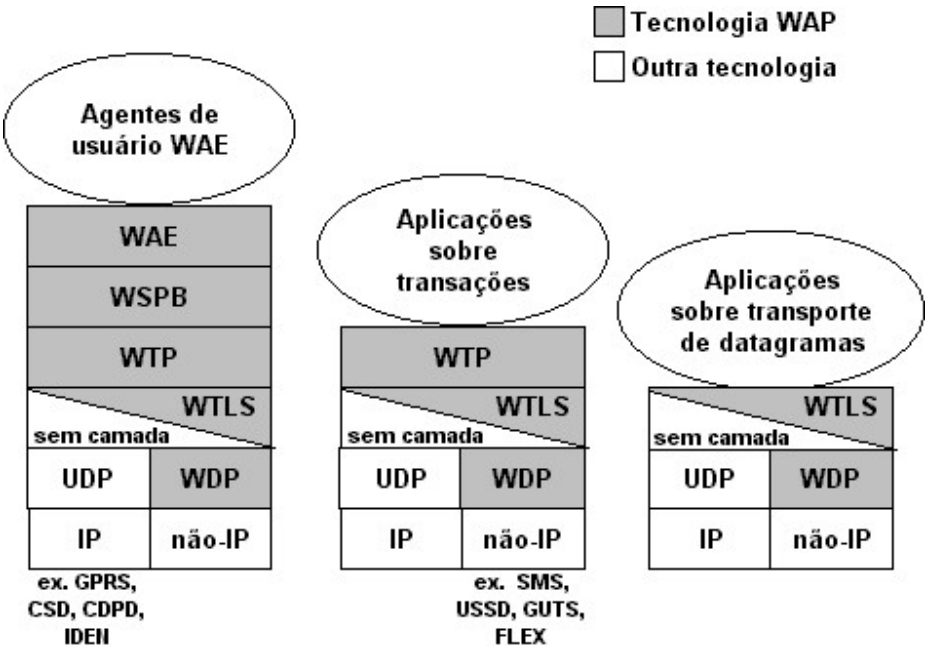

Figura 4.4 - Exemplos de pilhas WAP

A pilha mais à esquerda é um típico exemplo de uma aplicação WAP, isto é, um agente WAP rodando sobre a completa pilha de protocolos da tecnologia WAP. A localizada no meio é destinada a aplicações que requerem serviços de transação com ou sem segurança, e a pilha mais a direita é destinada a aplicações e serviços que somente requerem o transporte de datagramas com ou sem segurança.

4.9 WAP *Push*

A estrutura do WAP *Push* introduz uma maneira, dentro do esforço WAP, para transmitir informação ao dispositivo sem uma anterior ação do usuário. No conhecido modelo cliente/servidor, um cliente requisita um serviço ou informação a um servidor, que então responde transmitindo a informação ao cliente. Isto é conhecido como tecnologia "*pull*": o cliente puxa (*pull*) a informação do servidor (fig. 4.5). A WWW é um exemplo típico de tecnologia *pull*, onde um usuário entra com uma URL (a requisição) que é enviada ao servidor, e o servidor responde pelo envio de uma página *Web* (a resposta) ao usuário.

Em contraste disto, há a tecnologia *push*, que também é baseada no modelo cliente/servidor, mas onde não há uma requisição explícita do cliente antes do servidor transmitir o conteúdo.

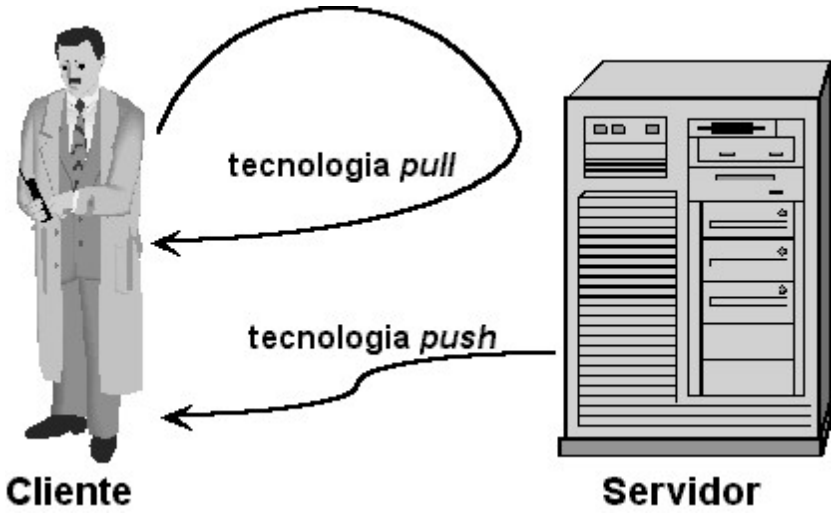

Figura 4.5 - Diferença de fluxo entre as tecnologias *pull* e *push*

Outra forma de explicar isso é que enquanto as transações "*pull*" da informação são sempre originadas pelo cliente, as transações "*push*" são originadas pelo servidor.

4.9.1 A Estrutura do WAP *Push*

Uma operação *push* em WAP ocorre quando um originador *push* envia o conteúdo ao cliente utilizando um dos seus protocolos. Nesta forma simples a arquitetura pareceria como a fig. 4.6.

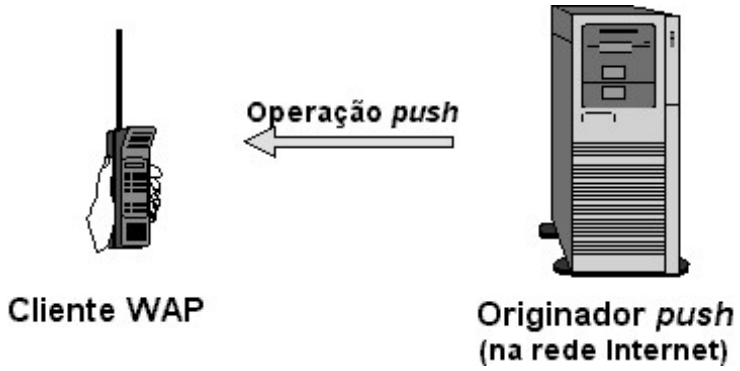

Figura 4.6 - Arquitetura *push* básica

Contudo, o originador *push* não compartilha nenhum protocolo com o cliente WAP: o originador *push* está na Internet e o cliente WAP no domínio WAP (fig. 4.7). Por isso, o originador *push* não pode se comunicar com o cliente WAP sem um mecanismo intermediário, então nós vamos precisar inserir um *gateway* para fazer essa tradução.

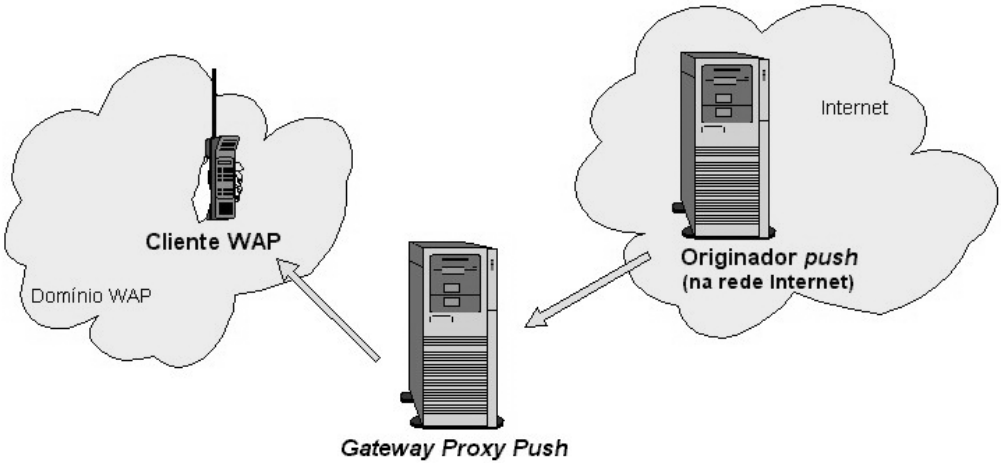

Figura 4.7 - Arquitetura *push* utilizando o *Gateway Proxy Push*

Assim, o originador *push* contata o *Gateway Proxy Push* (PPG) do lado da Internet, despachando conteúdo para o cliente destino utilizando protocolos da Internet. O PPG faz o que é necessário para repassar o conteúdo "empurrado" para o domínio WAP, e o conteúdo é transmitido pela rede de terminais móveis até o cliente destino.

Adicionalmente, para prover simples serviços de *proxy gateway*, o PPG pode estar encarregado de notificar o originador *push* sobre o final do resultado da operação de *push*, e ele pode esperar pelo cliente para aceitar ou rejeitar o conteúdo em redes móveis de dois sentidos. Isto também pode proporcionar ao originador *push*, juntamente com o cliente, serviços de procura, deixando o originador *push* selecionar as preferências de conteúdo especial para este cliente particular.

No lado da Internet, o protocolo de acesso ao PPG é chamado de protocolo de acesso *push*. No lado WAP, o protocolo é chamado de "*Push Over-the-Air*". Dessa maneira, um esquema revisado parece alguma coisa como mostrado na fig. 4.8:

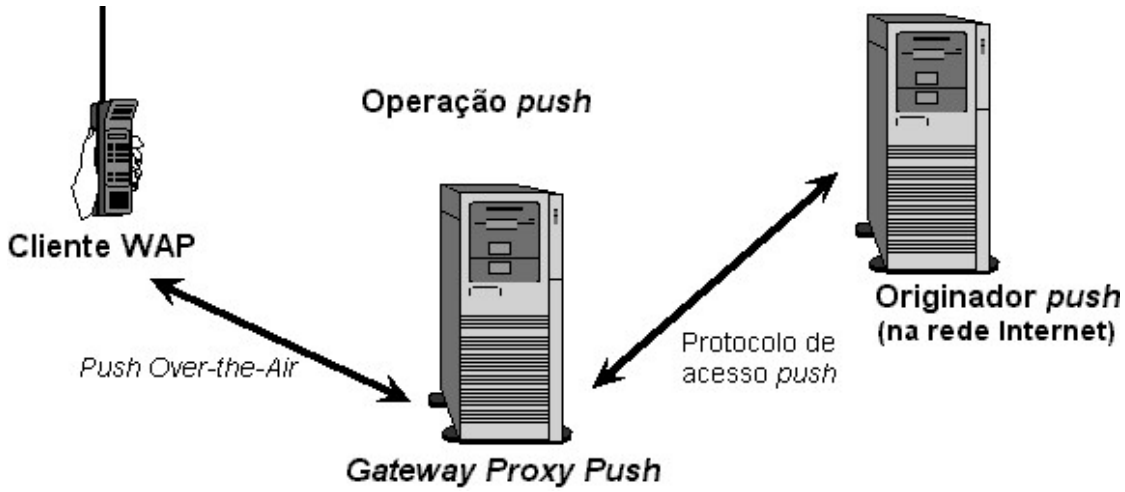

Figura 4.8 - Arquitetura *push* com os protocolos

O protocolo de acesso *push*, ou PAP, utiliza mensagens XML que podem estar embutidos em vários protocolos utilizados na Internet como o HTTP.

4.10 WML e WML*Script*

A WML [WAP 00] é uma linguagem descritora de documentos baseada em *tags*, assim como HTML. Embora tenha herdado aspectos de HTML [RAG 97] e HDML [KIN 97], é na realidade bem mais flexível, por tratar-se de um tipo de XML [BRA 98a], que é um conjunto simplificado da metalinguagem SGML. Ela é utilizada para especificar conteúdo e uma interface de usuário para dispositivos WAP considerando a limitada capacidade dos dispositivos como telefones celulares e outros terminais móveis sem fio. A linguagem WML diferencia maiúsculas de minúsculas, assim como a linguagem C. Conforme já mencionado, WML também foi derivada de HDML (*Handheld Device Markup Language*), uma linguagem proprietária da empresa Phone.com e comercializada pela mesma antes da criação do padrão WAP.

A XML consiste em uma metalinguagem, a qual é utilizada para definir outras linguagens, assim como o que foi feito para criar a linguagem WML. Ela permite ao desenvolvedor definir qualquer conjunto de *tags*, que são comandos inseridos em um documento, que servem para especificar como este documento (ou parte dele) deve ser apresentado. Um conjunto de *tags* é agrupado em um conjunto de regras de gramática e descreve uma definição de tipo de documento (DTD – *Document Type Definition*). O WAP *Forum* estabeleceu a DTD WML v1.1 que define sintaxe e elementos usados em um

documento WML válido. Esta DTD encontra-se disponível em http://www.wapforum.org/DTD/wml_1.1.xml.

Se um telefone móvel é dito como "WAP *Compatible*" significa que ele possui uma porção de *software* embutido, conhecido como *microbrowser*, que é capaz de interpretar todas as entidades dispostas em um documento WML.

A linguagem WML apresenta uma variedade de características, incluindo:

- Suporte para textos e imagens – WML provê aos autores (desenvolvedores) meios de especificar texto e imagens para serem apresentados ao usuário. Isto pode incluir *layout* e dicas de apresentação. Como outras linguagens de marcação, WML requer do autor especificar a apresentação em termos muitos gerais e dá ao agente de usuário uma grande liberdade para determinar exatamente como a informação é apresentada ao usuário final;
- Suporte para dados de entrada do usuário – WML suporta vários elementos para solicitar entrada de dados do usuário. Todas as requisições para entrada de dados são feitas em termos abstratos, permitindo que o agente de usuário tenha liberdade para otimizar características para cada dispositivo particular. Os campos de entrada de textos podem ter máscaras para impedir que o usuário insira tipos de caracteres inválidos;
- Navegação e pilha de histórico – WML permite vários mecanismos de navegação utilizando URLs. Ele também expõe um mecanismo de histórico. Navegação inclui *hiperlinks* no estilo HTML, elementos de navegação entre *cards* (v. seção 4.11.1), bem como histórico de elementos de navegação;
- Suporte a internacionalização – O conjunto de caracteres de documento para WML é o *Universal Character Set* da ISO/IEC-10646. Atualmente, este conjunto de caracteres é igual ao Unicode 2.0. Não há necessidade de que os *decks* (v. seção 4.11.1) WML sejam codificados usando a codificação completa do Unicode (exemplo, UCS-4). Qualquer conjunto de caracteres que seja um subconjunto correto de caracteres do Unicode pode ser usado (exemplo, US-ASCII, ISO-8859-1, UTF-8, Shift_JIS, etc.);

- Independência de interface homem-máquina – A especificação abstrata do *layout* e da apresentação do WML permite que fabricantes de dispositivos controlem a forma da interface homem-máquina para seus produtos, com cada um apresentando determinadas particularidades.
- Otimização para banda estreita – WML inclui uma variedade de tecnologias para otimizar a comunicação em um dispositivo de banda estreita. Isto inclui a possibilidade de especificar várias interações de usuário (*cards*) em uma transferência na rede (um *deck*);
- Gerenciamento de estado e contexto – Cada controle de entrada em WML pode introduzir variáveis. O estado das variáveis pode ser usado para modificar o conteúdo de um *card* parametrizado sem ter que haver comunicação com o servidor. Além disso, o tempo de vida do estado de uma variável pode ser maior que um único *deck* e pode ser compartilhado por vários sem ter que utilizar um servidor para salvar um estado intermediário entre as invocações dos *decks*.

4.10.1 *Decks* e *Cards*

A linguagem WML foi estruturada com o objetivo de utilizar a baixa taxa da transferência das redes sem fio e as pequenas telas dos dispositivos móveis de forma otimizada. Com esse objetivo, conceitos como *deck* e *card* foram utilizados. Um documento WML simples, que apresenta elementos contidos dentro de um par da *tag* "<wml>", é chamado de *deck*, e representa uma aplicação ou serviço. O *deck* é o primeiro elemento a ser lido pelo *microbrowser* e é nele que estão contidas as informações sobre a página, que são utilizadas para, por exemplo, identificar se a mesma pode ser visualizada pela versão do *microbrowser* disponível no dispositivo móvel.

Cada *deck* contém um ou vários *cards* (ou para melhor entendimento pode-se chamar de 'cartões'), os quais podem ser definidos como sendo as telas de informação a serem visualizadas pelo usuário. O *card* pode conter texto, imagem, referências para outros sítios ou campos de entrada.

O *deck* é similar a um documento HTML, uma vez que é identificada por um URL [BER 98]. Dessa forma, um *deck* é uma unidade de conteúdo de transmissão que é enviada integralmente do servidor WAP ao dispositivo móvel do usuário. Isto significa que várias telas (ou seja, *cards*) podem ser enviadas a um cliente de uma única vez. Um *deck* não

deve exceder 1400 *bytes*, o que não corresponde ao tamanho do arquivo WML e sim ao tamanho do arquivo binário codificado para ser transmitido, resultante do arquivo fonte WML.

Quando se desenvolve uma página para estar disponível na Internet, normalmente o arquivo que contém as instruções é salvo com a extensão HTML. Já quando se deseja que um determinado conjunto de informações seja acessado via WAP, a extensão do arquivo geralmente usada é WML.

As imagens possuem um formato especial chamado WBMP (*Wireless Bitmap*), que correspondem a imagens em preto e branco com tamanho de até 150 x 150 *pixels* – apesar de imagens coloridas também serem aceitas. Arquivos de imagens nos formatos JPEG, GIF ou outros utilizados na *Web* não podem ser diretamente referenciados em arquivos WML, necessitando serem convertidos para o formato WBMP, para que possam ser visualizados pelo *microbrowser* disponível no dispositivo móvel. Essa tarefa pode ser realizada de forma simples e rápida através da utilização de diversos programas gratuitos disponíveis, na Internet além de serviços de conversão *on-line* como o oferecido pela empresa Teraflops LTD. (http://www.teraflops.com/wbmp/). Além disso, como nem todos os terminais WAP têm capacidade de exibir imagens, pode-se utilizar a opção 'alt' (da mesma forma como é utilizado em um código HTML) para que esses terminais mostrem uma legenda no lugar da imagem.

4.10.2 WML*Script*

A WML*Script* [WAP 00b] é uma linguagem de *script* procedimental, semelhante ao Java*Script*, e é utilizada em conjunto com arquivos no formato WML. Ela estende as capacidades de navegação e apresentação do WML com características comportamentais, suporte para um comportamento mais avançado da interface de usuário, adiciona recursos inteligentes ao dispositivo móvel, provê um mecanismo inteligente para acessar o dispositivo e seus periféricos e reduz as necessidades de tráfego ao servidor de origem. A WML*Script* é baseada num subconjunto de instruções da linguagem de *scripts* Java*Script* a qual é bastante utilizada na WWW. Tem a finalidade de criar um meio padrão para adicionar lógica procedimental aos *decks* WML. Em geral, WML*Scripts* são funções criadas por provedores de conteúdo para, por exemplo, verificar se todos os campos de entrada de um *card* estão preenchidos. Essas funções podem ser invocadas por nomes, aceitam parâmetros e retornam valores. Com a utilização desses *scripts*, seleções do usuário

(ou entradas) podem ser manipuladas e enviadas a *cards* carregados, o que elimina a excessiva troca de dados entre o cliente e o servidor remoto.

Algumas das capacidades do WML*Script*:

- Habilidade para validar os dados de entrada do usuário antes de enviar ao servidor de conteúdo;
- Capacidade para acessar as facilidades do dispositivo e seus periféricos;
- Habilidade de interação com o usuário sem causar sobre carga de tráfego ao servidor de origem (ex. mostrar uma mensagem de erro).

Além destas capacidades, WML*Script* apresenta também outras características interessantes, como a possibilidade de o código WML*Script* ser compilado para diminuir a quantidade de *bytes* a serem transferidos do servidor ao cliente, e por ser o WML*Script* uma linguagem baseada em eventos, existe a possibilidade de invocar um *script* para interagir com o usuário ou reagir a eventos de ambiente.

4.10.3 Acesso a Conteúdo WML e WML*Script*

A fig. 4.9, a seguir, apresenta as diferentes partes da arquitetura lógica assumida por um agente de usuário WML.

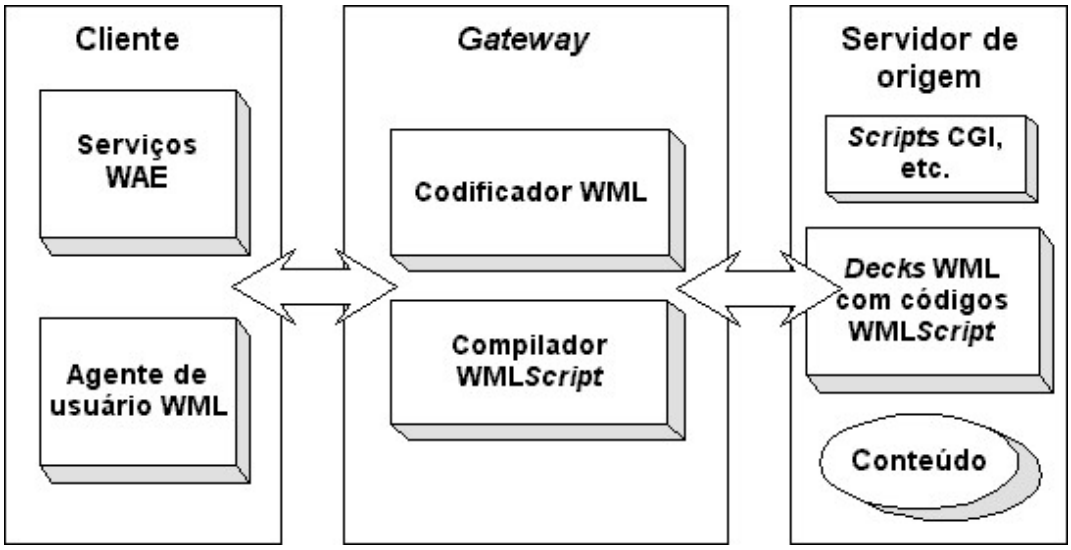

Figura 4.9 - Arquitetura lógica do agente de usuário WML

Servidores de origem provêm serviços de aplicação ao usuário final. O serviço de interação entre o usuário final e o servidor de origem é empacotado como *scripts* e *decks* WML. Serviços podem contar com *decks* e *scripts* que estão armazenados estaticamente

no servidor de origem, ou podem contar com conteúdo produzido dinamicamente por aplicações no servidor de origem.

Alguns estágios estão envolvidos quando agentes de usuário WML e servidores de origem trocam informações no formato WML e WML*Script*. Em especial, um usuário, desejando acessar um serviço qualquer, envia uma requisição ao servidor de origem utilizando um agente de usuário WML. O agente de usuário requisita o serviço ao servidor de origem em nome do usuário utilizando, alguma operação do esquema URL (ex. método de requisição GET do protocolo HTTP).

O servidor de origem então valida a requisição do usuário e como resposta envia o conteúdo de um *deck*. Presumidamente este *deck* está inicialmente em um formato textual. No seu caminho de ida ao cliente, os *decks* textuais passam por um *gateway* onde são convertidos para formatos mais adaptados para transmissões sem fio e dispositivos de processamento limitado. Em princípio, uma vez que o *gateway* recebe o *deck* do servidor original, o *gateway* realiza todas as conversões necessárias entre o formato de texto e binário. Um codificador WML contido no *gateway* converte cada *deck* WML para seu respectivo formato binário. O conteúdo codificado é enviado ao cliente para ser visualizado e interpretado. As requisições por conteúdo WML*Script* são tratadas por um compilador WML*Script* que compila o *script* em um *bytecode* (otimizado para baixa largura de banda) e envia ao cliente para que seja interpretado e executado.

Não é obrigatória a presença de um *gateway* nessa estrutura, podendo os dispositivos de codificação e compilação estarem disponíveis de outra forma. Pode-se, por exemplo, adicionar estas duas funções ao próprio servidor de origem, como é ilustrado na fig. 4.10, onde o conteúdo já se encontra codificado em *bytecode*, o que elimina a necessidade de qualquer conversão.

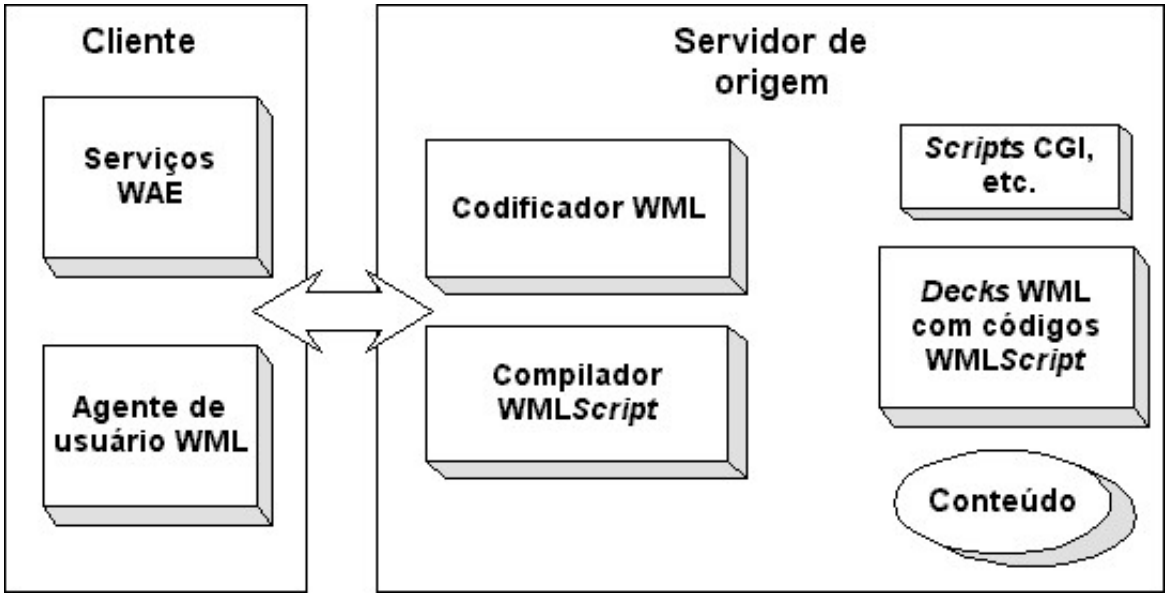

Figura 4.10 - Arquitetura lógica do agente de usuário WML (sem a utilização do g*ateway*)

4.10.4 Segurança e Controle de Acesso

A camada de aplicação WAE influencia o protocolo de segurança WTLS onde serviços requerem trocas de informações autenticadas ou seguras define a biblioteca de interface para WML*Script* e provê funcionalidade de criptografia aos agentes de usuário WAE [WAP 99a]. Adicionalmente, WML e WML*Script* incluem instruções de controle de acesso que comunica ao cliente as restrições no acesso baseado em URL. Essas construções permitem que autores de conteúdo WML e WML*Script* garantam acesso público a determinado conteúdo (ex. um *deck* ou *script* que pode ser referenciado por outro conteúdo) ou restringir o acesso ao conteúdo pelo ajuste de *decks* ou *scripts* "confiáveis". O protocolo WAE também suporta a autenticação básica do protocolo HTTP v1.1 onde um servidor pode autenticar o usuário baseado em um nome de usuário e senha.

4.10.5 Exemplo Básico

Abaixo é apresentado um documento WML (fig. 4.11), representado por um *deck* e contendo um único *card*. Sua função é a de somente imprimir na tela a mensagem "Minha Primeira página WAP". Devem ser observadas com atenção as três primeiras linhas do documento, que o definem como um arquivo WML e especificam qual é sua versão – além da versão da linguagem XML. Estas três linhas devem formar o cabeçalho de qualquer documento WML.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card>
<p> Minha Primeira pagina WAP </p>
</card>
</wml>
```

Figura 4.11 - Exemplo de deck WML

A próxima linha corresponde à declaração do início do *deck*. As *tags* "`<wml>`" e "`</wml>` " delimitam todo o conjunto de sub-elementos e entidades que formam o código WML. Cabe ressaltar que cada um desses sub-elementos deve estar entre os caracteres "`<`" e "`>`". Cada sub-elemento deve ter seu início e fim demarcados, na forma `identificador` e `/identificador`, respectivamente (p.ex. `<card>` ... `</card>`). Dessa forma percebe-se que entre as linhas cinco e sete foi definido o *card* do documento WML.

Os comentários podem ser identificados por estarem delimitados pelo conjunto de caracteres "`<!-`" e "`->`" ou "`/*`" e "`*/`" que marcam o início e fim do bloco de comentários. Duas barras "`//`" também definem que o resto da linha é comentário.

Outros exemplos podem ser consultados no trabalho de Dias [DIA 00], que apresenta também outras informações como tipos de variáveis e operadores utilizados em *scripts* WML e WML*Script*.

5 O *Framework* de Aplicações VIGIA

Um *framework* é um esqueleto para uma família de aplicações relacionadas, desenhado para ser reusado. É composto por um conjunto de classes que definem uma solução genérica para um domínio de aplicação. Algumas dessas classes são muitas vezes abstratas, pois implementam o comportamento genérico do domínio deixando alguns métodos sem implementação. Esses métodos (chamados métodos abstratos) devem ser implementados pelas subclasses específicas, que definem variações diferentes do comportamento genérico. Dessa forma, o usuário de um *framework* produz uma nova aplicação através de um processo que envolve duas fases:

1. Especializar as classes abstratas para implementar a funcionalidade específica necessária pela aplicação (se ainda não existir na biblioteca uma classe que satisfaça essa funcionalidade);
2. Descrever como as instâncias dessas classes são interconectadas para implementar a totalidade da aplicação.

Este capítulo descreve o *framework* VIGIA, que foi projetado para servir de arcabouço para o desenvolvimento de aplicações de monitoração, alerta e intervenção remotos que utilizem as tecnologias de comunicação descritas anteriormente. Começa fazendo uma descrição dos princípios de funcionamento de tais aplicações, para logo a seguir iniciar a descrição do *framework* propriamente dito.

5.1 Perspectiva Conceitual

Segundo as recomendações de Steve Cook e Jonh Daniels [apud FOW 00], o projeto de qualquer modelo pode ter sua execução facilitada e resultados mais facilmente compreendidos através da adoção de uma seqüência baseada em três perspectivas: a Conceitual, a de Especificação e a de Implementação. Esta seção aborda a perspectiva conceitual, descrevendo o comportamento, recursos providos e ambiente de utilização do *framework* VIGIA. Numa seção posterior, serão abordadas as outras duas perspectivas.

5.1.1 Descrição Funcional

O *framework* VIGIA fornece a estrutura e os mecanismos básicos para o funcionamento de um sistema genérico de monitoração, alerta e intervenção remotos, que tem seu funcionamento descrito a seguir:

Da Monitoração

Um sistema desenvolvido a partir do *framework* VIGIA, monitora uma série de aspectos do ambiente que ajuda a manter (vigiar), através de um conjunto de sensores que estão constantemente informando valores sobre os objetos monitorados. Cada objeto pode ter um ou mais sensores, cada qual voltado a medir uma grandeza. Esses sensores podem ser concretos, medindo grandezas físicas tais como temperatura (neste caso é um termômetro), ou abstratos, medindo grandezas mais elaboradas como a vazão numa determinada interface de rede. No primeiro caso, a leitura poderia ser expressa em graus Celsius (°C); no segundo, poderia ser expressa em *kilobytes* por segundo (kBps), por exemplo.

Uma vez feita a coleta de todos os valores que estão sendo monitorados, os mesmos são verificados e comparados a limites, para determinar se existe alguma anormalidade digna de nota. Essa avaliação é feita através de um conjunto de expressões lógicas e aritméticas, que podem envolver um ou mais valores lidos.

Do Alerta

Quando a avaliação de uma dessas expressões resulta num valor *booleano* (lógico) verdadeiro, diz-se que atingiu uma "condição de alerta". A cada condição de alerta disparada, está associada uma mensagem específica, além do horário em que foi detectada e o intervalo mínimo entre disparos (ou notificações), entre outros atributos.

No caso de existir uma condição de alerta que deva ser notificada (a primeira, ou após o intervalo mínimo desde a última), a mensagem associada é enviada a um ou mais telefones celulares (ou dispositivos móveis quaisquer.

Quando isso ocorre, dependendo da natureza da condição de alerta e da sua gravidade (severidade), pode ser iniciado um cronômetro que disparará nova condição de alerta, depois de transcorrido um certo tempo desde o envio da mensagem sem que tenha ocorrido uma intervenção humana.

Essa nova condição de alerta pode gerar ações do mesmo tipo da anterior, ou seja, fazer nova tentativa de notificação, possivelmente com outro(s) destinatário(s), ou

ainda comandar operações automatizadas, como o desligamento parcial ou total do sistema sendo monitorado.

Da Intervenção

O sistema mantém uma lista de usuários associados a cada objeto sendo monitorado. Tais usuários podem interagir com o sistema através de uma interface *Web*. Para iniciar uma sessão de trabalho, o usuário precisa ser identificado e autenticado. Assim, o sistema sabe a qual(is) objeto(s) o usuário está vinculado, apresentando informações mais detalhadas sobre estes e permitindo que o usuário comande ações alteradoras (intervenha) sobre tais objetos. Os usuários podem interagir com o sistema em resposta a mensagens recebidas ou por iniciativa própria. No primeiro caso, o cronômetro acima referido é desligado (caso tenha sido iniciado).

Do Registro de Atividades

As principais ocorrências relativas aos estados de alerta e correspondentes envios de mensagens, bem como das eventuais intervenções executadas pelos usuários, devem ser registradas. Além dessas ocorrências, quaisquer outras atividades que alterem o estado do sistema, a critério do administrador, também podem ser registradas. Esse registro permitirá recuperar o histórico de funcionamento do sistema, podendo ou não ser seletivo (apenas sobre um conjunto de objetos monitorados, por exemplo). Nesse tipo de sistema, registros históricos são essenciais para a realização de auditorias e verificação do correto funcionamento do sistema.

5.1.2 Cenários de Aplicação

O *framework* VIGIA foi projetado tendo em vista uma série de aplicações típicas de vigilância ou monitoração, sendo o resultado de várias idéias sobre campos de atuação.

Dentre estes se destacam as aplicações para a área médica, onde já existem diversas iniciativas com objetivos complementares, a ponto de configurar uma nova área de pesquisa e desenvolvimento denominada “telemedicina”. Outro campo onde figuram aplicações semelhantes, que apenas enviam alertas sem permitir a intervenção, é a área de gerência de redes [REG 00, YUR 00].

Nesta seção, serão descritas algumas das possíveis aplicações construídas a partir do *framework* VIGIA, com o objetivo de ressaltar seu potencial mercadológico.

Gerência de Redes de Computadores

Muitas das redes locais de computadores necessitam estar em pleno funcionamento constantemente, indiferentes a períodos de descanso humano. Essa indiferença se reflete também quando "escolhem" os momentos mais inoportunos para apresentarem problemas de mau funcionamento.

Como a maioria das redes local já possui conexão permanente à Internet, além das facilidades de gerência de redes já existentes, são candidatas naturais à adoção de um sistema de monitoração, alerta e intervenção remotos.

Uma rede local de computadores geralmente é configurada para o uso de um sistema de gerência qualquer, muitas vezes baseado no protocolo SNMP (*Simple Network Management Protocol*), por exemplo.

Pois bem, para a adoção de um sistema de vigilância do tipo proposto, bastaria que se configurassem alguns agentes SNMP como sensores e atuadores desse sistema, tarefa muito simples.

A partir daí, a equipe de administradores de rede poderia ser notificada a qualquer hora e em qualquer lugar coberto pela sua operadora de telefonia celular, sobre quaisquer situações de anormalidade que quisessem. Também poderiam, sem se deslocar do lugar, tomar atitudes corretivas para restaurar o bom funcionamento da rede.

Sistemas de Segurança Domiciliar

A proteção de residências também é uma área que tem concentrado muita atenção, pois os prejuízos decorrentes de diversos tipos de sinistros são consideráveis, sem falar nos riscos à vida que alguns sinistros implicam.

Um sistema de vigilância segundo o modelo proposto poderia ser utilizado para melhorar a segurança domiciliar em vários tipos de estruturas residenciais, como condomínios horizontais, verticais e residências isoladas.

As residências deveriam estar providas de diversos tipos de sensores, como sensores de fumaça para princípios de incêndio, detectores de movimento e sensores para detectar o estado da aberturas, entre outros.

Assim, quando tais sensores detectassem situações de invasão ou perigo iminente, seriam emitidas mensagens urgentes alertando os proprietários e/ou responsáveis pela segurança das residências. Essas situações podem ser facilmente configuradas e adaptadas a cada política de segurança adotada pelos proprietários.

Por exemplo, num condomínio vertical, o sistema poderia ser facilmente configurado pelo administrador para alertar todos os moradores dos andares vizinhos em caso de indícios de fumaça num determinado apartamento.

As intervenções dos proprietários estariam limitadas apenas pelo tipo de dispositivos que as residências dispusessem, sendo que os mais evidentes permitiriam ligar ou desligar equipamentos, trancar ou destrancar portas e janelas e acionar dispositivos de alarme (por exemplo, no distrito policial).

5.2 Especificação e Implementação do *Framework*

Após a análise detalhada da descrição funcional genérica (exposta acima) de um sistema com os propósitos almejados, foram identificados os aspectos comuns, que independem das particularidades de cada sistema. Esses aspectos referem-se, principalmente, aos processos de monitoração de objetos, envio de mensagens de alerta de forma imediata aos usuários e interface de interação e intervenção sobre o sistema/ambiente monitorado.

Tais aspectos comuns definem o comportamento genérico do sistema, ou seja, podem ser implementados por um conjunto de classes que formam o *framework* desejado. Para dar início à descrição do *framework*, serão apresentadas suas cinco principais classes (fig. 5.1):

1. **Vigia** - Esta classe é a representante principal de um sistema construído a partir do *framework*, sendo responsável pela integração e coordenação de todas as atividades relativas ao funcionamento desse sistema de vigilância;
2. **Monitor** - É a classe reponsável pela observação das condições do ambiente monitorado e os conseqüentes disparos de situações de alerta;
3. **Alerta** - As instâncias desta classe têm por finalidade fazer o envio das mensagens de alerta aos usuários do sistema;
4. **Atuador** - A partir desta classe são definidos os possíveis tipos de intervenção remota sobre o ambiente monitorado que o sistema admite;
5. **Registro** - Esta é a classe responsável por realizar o registro de todos os eventos e atividades dignos de nota.

Um sistema com os propósitos em questão pode ser composto por diversas instâncias da classe **Monitor** (diversos monitores), cada qual associado a um objeto a ser monitorado. O sistema pode também estar atendendo, simultaneamente, a mais de um usuário que deseja intervir no ambiente monitorado (diversas instâncias de **Atuador**). A fig. 5.1 apresenta o diagrama em UML que mostra as relações entre as cinco principais classes do *framework*.

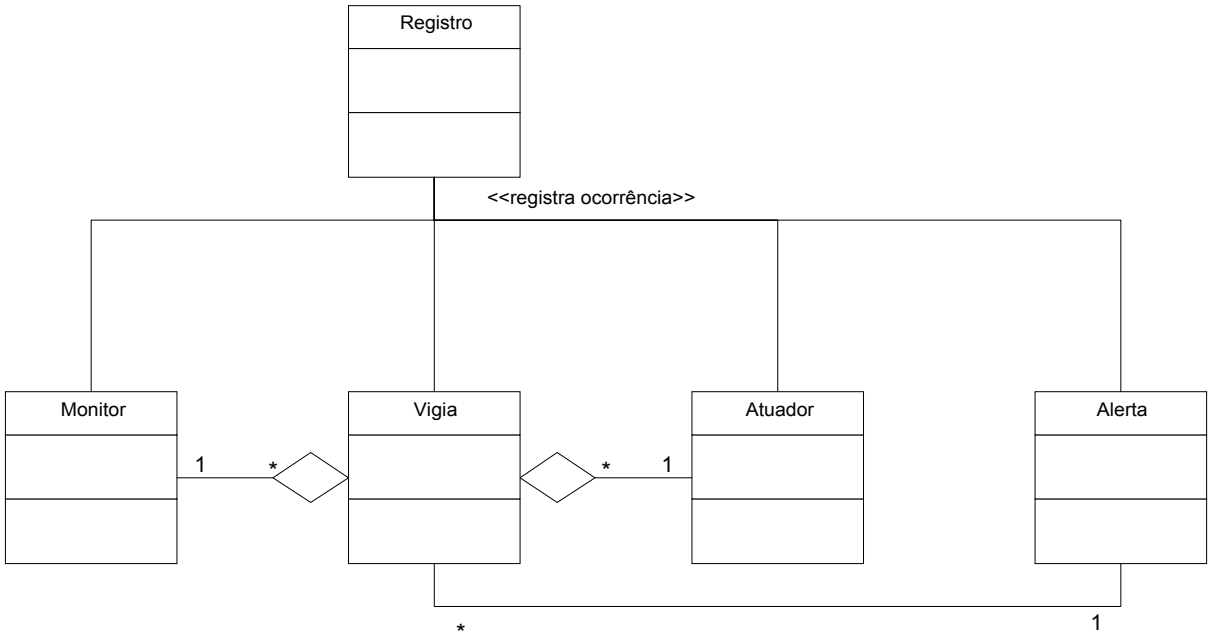

Figura 5.1 - Diagrama das principais classes do *framework* VIGIA

5.2.1 Ambiente de Funcionamento

O *framework* VIGIA foi projetado para ser independente dos meios de comunicação empregados. Conforme descrito acima, as necessidades de comunicação com o meio externo são decorrentes de duas atividades: o envio de alertas e o acesso via *Web* para permitir a intervenção. São apenas identificadas tais necessidades de comunicação, sem entretanto especificar como essa comunicação será realizada.

Conseqüentemente, uma aplicação construída com o *framework* VIGIA necessita de uma série de recursos de comunicação para o seu funcionamento. Por exemplo, para o envio dos alertas, poderia ser utilizada a tecnologia WAP *Push*, descrita na seção 4.x.x.

Um dos requisitos básicos do *framework* é permitir uma alta mobilidade aos usuários. Tal mobilidade é encontrada na rede de telefonia celular, que hoje na sua versão digital, permite uma grande integração com a Internet.

A tecnologia de comunicação de dados via rede celular digital, disponível na nossa região, restringiu a escolha dos meios de comunicação para a implementação e teste do *framework* VIGIA. Para o envio de mensagens sem demanda, as operadoras da nossa região só oferecem o serviço SMS, e para o acesso à Internet, só está disponível o protocolo WAP.

Outra característica importante é a necessidade de conexão permanente do sistema construído a partir do *framework* com a rede Internet, para permitir a comunicação direta e imediata com a operadora de telefonia e também permitir a comunicação entre o dispositivo móvel do usuário e o sistema.

A fig. 5.2 ilustra as necessidades de comunicação entre os diversos componentes de um sistema construído conforme o modelo proposto e algumas das alternativas possíveis de tecnologias e protocolos.

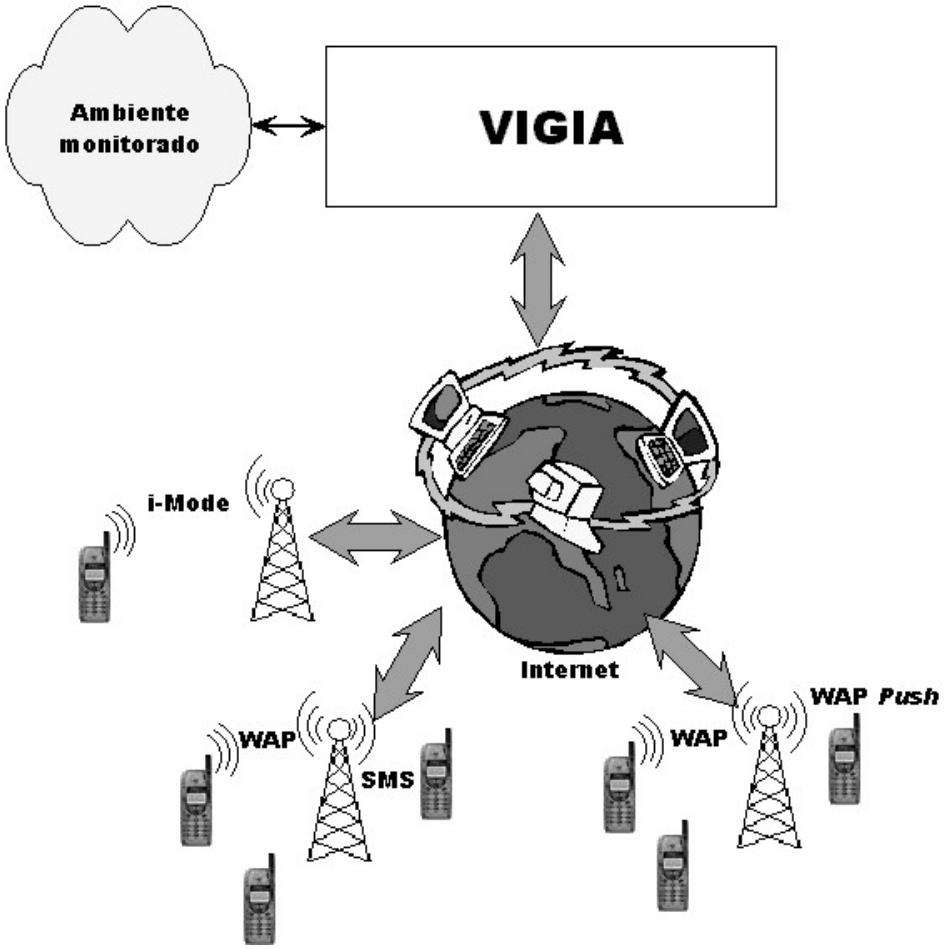

Figura 5.2 – Ambiente de funcionamento de um sistema de vigilância conforme VIGIA

5.2.2 Modelo Arquitetônico

O processo de desenvolvimento adotado na construção do *framework* VIGIA baseou-se na aplicação de algumas técnicas de análise e projeto orientados a objetos. O resultado desse processo pode ser visualizado na fig. 5.3, onde estão representadas as classes mais importantes do *framework*, agora expostas num nível de detalhamento que permite sua análise mais profunda.

Algumas das classes do modelo simplificado exposto anteriormente (fig. 5.1) foram decompostas, como é o caso da classe **Monitor**. Outras ainda tiveram algumas de suas responsabilidades transferidas para outras classes mais específicas, como demonstra o surgimento das classes **VigiaConfig** e **VigiaServlet**.

Outra técnica de projeto importante aplicada é a adoção de "interfaces" conforme o modelo Java [SUN 9?, COR 97, FOW 00], o que amplia a reusabilidade de classes simplificando o projeto e a implementação de novas classes. Na seção 5.2.2 a seguir, são descritos cada um dos componentes principais do modelo arquitetônico do *framework* VIGIA, onde a questão do uso de interfaces Java também é melhor explicada.

Na fig. 5.3, as classes hachuradas em cinza não fazem parte do *framework*, sendo apenas exemplos de classes que especializam as classes abstratas **Sensor**, **Alerta** e **Atuador** para uma aplicação específica; neste caso, a monitoração de objetos com controle de temperatura.

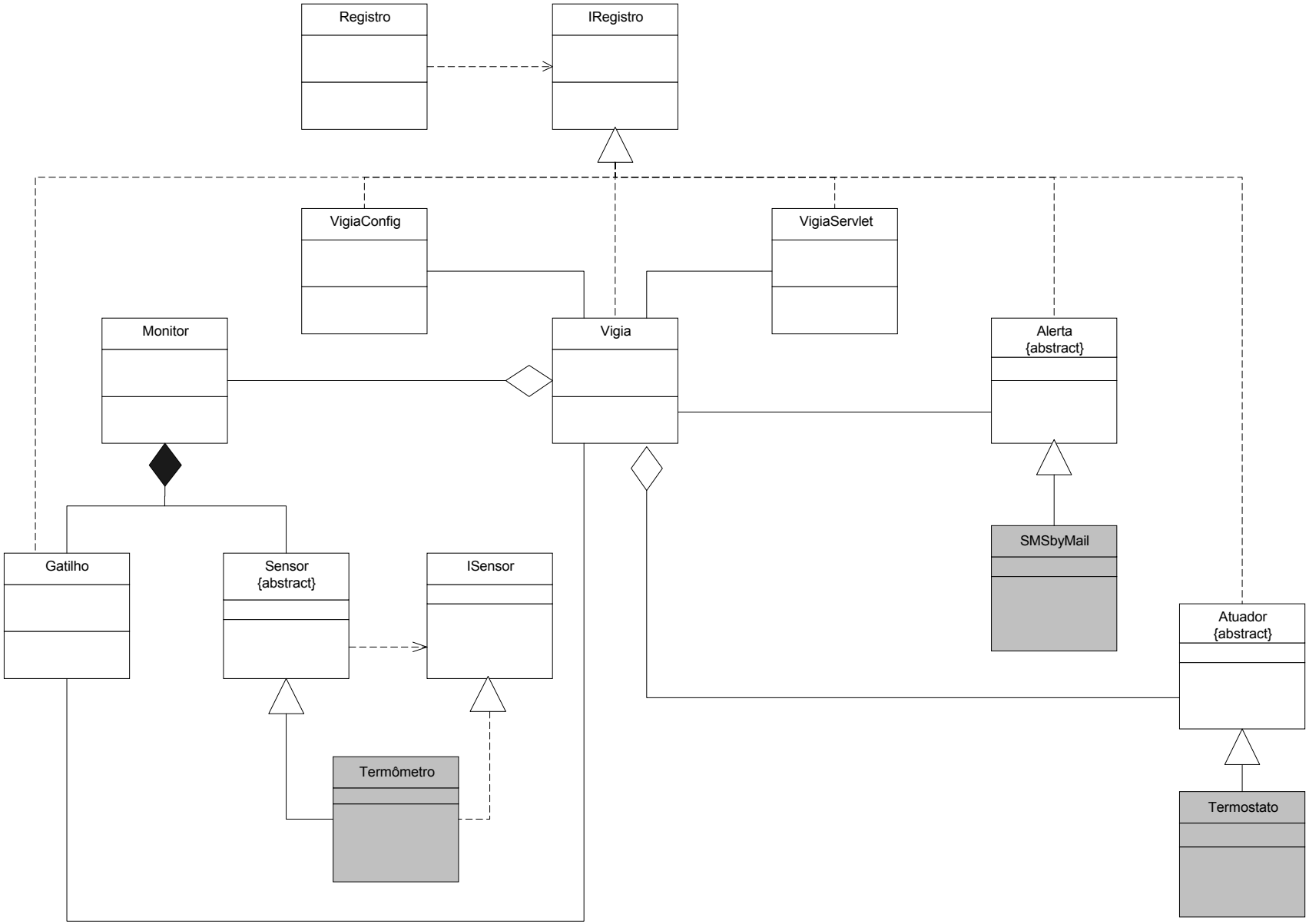

Figura 5.3 - Diagrama de classes do *framework* VIGIA

5.2.3 Descrição Geral dos Componentes

Nesta seção, é apresentada uma descrição estrutural e funcional das nove principais classes que compõem o *framework* VIGIA, conforme ilustradas na fig. 5.3.

Essa descrição busca dar uma visão geral do funcionamento de cada classe e suas formas de cooperação com as outras classes, fornecendo assim uma idéia mais precisa sobre o funcionamento de um sistema construído a partir desse *framework*. Além da descrição de suas responsabilidades, são apresentados seus principais atributos e métodos.

Para evitar repetições, é importante salientar neste ponto que todas as classes que implementam a interface **IRegistro** (Vigia, VigiaConfig, VigiaServlet, Gatilho, Alerta e Atuador, conforme fig. 5.3) têm a obrigação de implementar o método **registraEvento(Evento)**.

Classe Vigia

A classe **Vigia** (fig. 5.4) representa o "coração" do sistema baseado no *framework* VIGIA, sendo responsável pela coordenação das atividades dos outros objetos. Cada sistema possui uma única instância desta classe, que é implementada como um processo *daemon*, isto é, fica executando num laço constante em *background*.

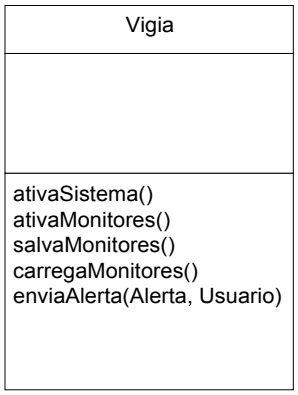

Figura 5.4 - A classe Vigia

Esta classe também implementa as funções administrativas do sistema, permitindo configurar e controlar o funcionamento deste. Entre seus principais métodos figuram:

- **ativaSistema()** - inicia a execução do processo *daemon* que representa a instância única desta classe;
- **desativaSistema()** - encerra a execução do processo *daemon*;
- **ativaMonitores()** - inicia a execução de todas as *threads* que representam instâncias de monitores;
- **desativaMonitores()** - encerra a execução de todas as *threads* que representam instâncias de monitores;
- **salvaMonitores()** - provoca o salvamento do estado atual de execução de todos os monitores, preparando para a interrupção do funcionamento do sistema;
- **carregaMonitores()** - recupera o estado de execução dos monitores, permitindo que o sistema seja reiniciado a partir do ponto em que foi interrompido;
- **ativaMonitor(Monitor)** - ativa individualmente um monitor, dando início à execução da *thread* correspondente;
- **desativaMonitor(Monitor)** - encerra a execução da *thread* correspondente ao monitor;
- **enviaAlerta(TipoAlerta, Monitor)** - cria uma instância da classe **Alerta** para o envio de uma mensagem de alerta, segundo o tipo designado, a um ou mais usuários associados ao monitor informado;
- **disparaAlerta(Alerta, Usuario, SeqAlerta)** - dispara um mecanismo de controle sobre a emissão de alerta com espera por resposta, cuja seqüência de ativação e limites de tempo são determinados pelo objeto SeqAlerta.

Classe VigiaConfig

Esta classe representa a interface administrativa do sistema, constituindo-se de uma interface gráfica que contém as informações sobre o estado atual do sistema, além de permitir sua configuração.

A classe **VigiaConfig** (fig. 5.5) admite uma única instância em execução a cada vez, por questões de segurança. Essa instância é iniciada (caso não haja outra) através da iniciativa do usuário administrador.

Uma vez iniciada, essa instância verifica a existência do processo *daemon* que representa a instância única de **Vigia**. No caso de existir, estabelece comunicação com esse processo, obtendo assim informações sobre o estado atual do sistema.

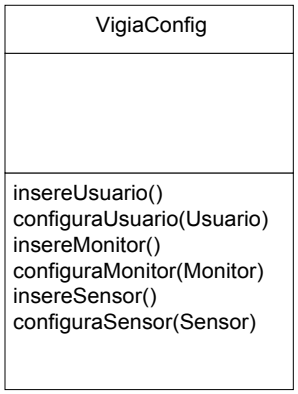

Figura 5.5 - A classe VigiaConfig

Como interface de controle sobre a execução do sistema, sua principal responsabilidade é fornecer uma interface de ativação dos métodos públicos da classe **Vigia**, já mencionados acima.

Para as tarefas de configuração do sistema, que permite ajustá-lo para as condições específicas do ambiente monitorado (vigiado), entre seus principais métodos estão:

- **insereUsuario()** - acrescenta novo usuário ao sistema;
- **configuraUsuario(Usuario)** - permite ajustar as propriedades do usuário, como seu endereço para encaminhamento de alertas (endereço para SMS), grupo a que pertence (seu papel no sistema), entre outras;
- **insereMonitor()** - acrescenta ao sistema um novo monitor, que estará vinculado a um objeto e/ou aspecto a ser vigiado;

- **configuraMonitor(Monitor)** - permite ajustar as propriedades do monitor, como número e tipo de sensores, regras de avaliação (gatilhos) dos alertas, entre outras;
- **insereSensor()** - acrescenta um novo sensor à lista de um determinado monitor;
- **configuraSensor(Sensor)** - configura as propriedades de um sensor, como o intervalo mínimo entre tomadas de valor, faixa de valores admissíveis, entre outras.

Classe VigiaServlet

Esta classe (fig. 5.6) é responsável pela interação com os usuários do sistema, representando a interface *Web* de uso que permite obter informações mais detalhadas a respeito do ambiente sob vigilância, além de poder comandar ações que modifiquem o estado desse ambiente.

Uma vez iniciada sua execução, busca comunicar-se com a instância única da classe **Vigia** para obter as informações que permitirão construir a interface de interação de acordo com a identificação do usuário. Caso não consiga comunicar-se, é indicação de que o sistema não está em execução, ou seja, não existe o processo *daemon* correspondente à instância da classe **Vigia**, por estar o sistema inativo. Neste caso, uma resposta relatando essa condição (de inatividade do sistema) é retornada.

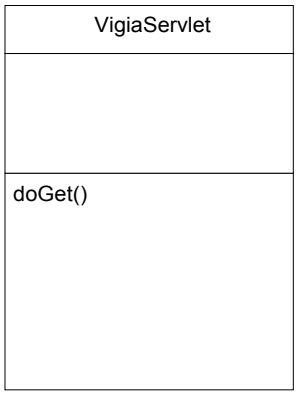

Figura 5.6 - A classe VigiaServlet

A classe **VigiaServlet** é uma especialização da classe HttpServlet originária do pacote de classes padrão da Sun [SUN 9?]. A classe HttpServlet implementa

uma API que permite aos servidores *Web* carregarem e executarem aplicativos Java, de forma semelhante ao que fazem os *browsers* ao executarem *applets*.

A fig. 5.7 mostra parte do código de implementação da classe **VigiaServlet**, onde percebe-se sua condição de especialização da classe HttpServlet e implementadora da interface **IRegistro**, explicada adiante.

```
public class VigiaServlet extends HttpServlet implements IRegistro {
...
public void doGet
      (HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      PrintWriter out;
      String title = "VIGIA - A qualquer hora, em qualquer lugar";
      response.setContentType("text/wml");
      out = response.getWriter();

      out.println("<?xml version="1.0"?>");
      out.println("<!DOCTYPE wml PUBLIC ...");
      out.println("<WML>");
      out.println(title);
      out.println("<card>");
      ...
      ...
      out.println("</card>");
      out.println("</WML>");
      out.close();
      }
}
```

Figura 5.7 - Listagem do método "doGet" da classe VigiaServlet

Classe Monitor

A classe **Monitor** (fig. 5.8) é uma composição de sensores e gatilhos, sendo responsável pela coordenação das atividades destes. É implementada como uma *thread*, permitindo assim que diversos monitores coexistam num mesmo sistema. Seu método principal é o ***run*()** (herdado da classe *Thread*), que é implementado como um laço permanente, conforme pode ser observado na fig. 5.9.

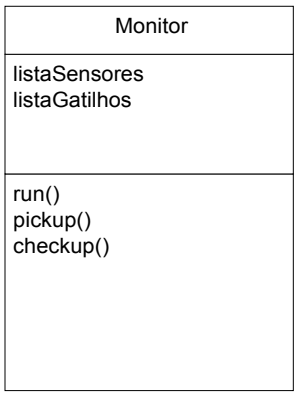

Figura 5.8 - A classe Monitor

Esse laço fica constantemente verificando o estado atual dos sensores, via chamada do método ***pickup*()**, e fazendo a análise da ocorrência de condições de alerta, através do método ***checkup*()**.

Outra particularidade digna de nota da classe **Monitor** é sua condição de implementadora da interface *java.io.Serializable*, o que permite que o estado de cada objeto seja armazenado e recuperado de disco, ou seja, são objetos persistentes.

```
package vigia;

public class Monitor extends Thread implements java.io.Serializable {
      private String monitorId;
      private int nrSensores;
      private Sensor[] listaSensores;
      private Gatilho[] listaGatilhos;
      ...

      public void run() {
            while(true) {
                  pickup();
                  checkup();
                  try {
                        sleep(1);
                  } catch (InterruptedException e) {yield();};
            };
      };

   ...

   private void pickup() {
      for(s= 0; s < nrSensores; s++)
         listaSensores[s].leValor();
   };

   ...

   private void checkup() {
      for(s= 0; s < nrSensores; s++)
         listaGatilhos[s].avalia(listaSensores[s]);
   };
}
```

Figura 5.9 - Listagem parcial da implementação da classe Monitor

Classe abstrata Sensor e Interface ISensor

A classe abstrata **Sensor** define o comportamento padrão para todos os sensores do sistema. Sua mais importante responsabilidade é fazer o controle dos intervalos de obtenção de valores dos sensores reais. Esse controle é realizado dentro do método **leValor()**, conforme se observa na fig. 5.11.

Como esse método genérico é implementado pela classe abstrata **Sensor**, suas especializações (com uma classe **Termometro**, por exemplo) devem possuir uma forma de integração com esse método. Tal forma é a implementação do método **mede()**, invocado dentro do método **leValor()**.

Aqui o mecanismo adotado para tornar esse vínculo forte é novamente a adoção de uma interface, denominada **ISensor**, que exige a implementação do método **mede()**, conforme se observa nas fig. 5.10, 5.11 e 5.12. Nesta última, figura está ilustrado apenas o cabeçalho de definição de uma hipotética classe **Termometro** e sua implementação do método **mede()**, cumprindo seu compromisso com a interface ISensor.

```
interface ISensor {
  private float mede();
      ...
}
```

Figura 5.10 - Listagem parcial da interface ISensor

```
public abstract class Sensor {
   ...

   public void leValor() {
      now= Date.miliseconds();
      if(now >= lastPickup + intervaloCheck)
         mede();
   }
   ...
}
```

Figura 5.11 - Listagem parcial da classe abstrata Sensor

```
public class Termometro extends Sensor implements ISensor {
   ...
   private float mede() {
      ...
      ...
   };
}
```

Figura 5.12 - Listagem parcial da classe Termometro

Classe Gatilho

A classe **Gatilho** (fig. 5.13) é responsável pela avaliação do conjunto de expressões (regras) que representam as condições de alerta. Um gatilho inicia sua avaliação pela busca do conjunto de valores a serem avaliados dentro das expressões, obtidos através de um pedido (invocação de método) ao objeto Sensor correspondente.

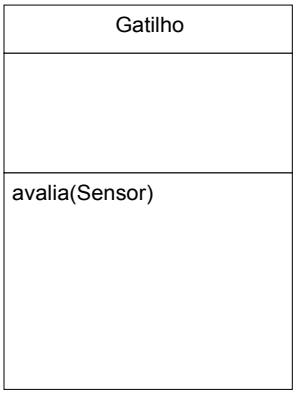

Figura 5.13 - A classe Gatilho

Após ter obtido esse conjunto de valores, os substitui nas expressões associadas àquele sensor e as avalia, uma a uma. Para cada resultado lógico verdadeiro, invoca o método **enviaAlerta(TipoAlerta, Monitor)** do objeto Vigia.

O decompositor e analisador de expressões (*parser*) utilizado é proveniente do pacote *nFunk.parser*, que originalmente admitia apenas expressões matemáticas, tendo sido estendido para avaliar expressões lógicas também (v. Anexo 2).

Essas expressões são fornecidas pelo administrador do sistema, sendo armazenadas na forma de um literal (*string*). A sintaxe dessas expressões deve seguir a sintaxe da linguagem Java, utilizando um subconjunto de operadores aritméticos e lógicos dessa linguagem (v. tabela 5.1).

As expressões podem fazer uso também de funções e constantes, que podem ser facilmente estendidas pelo projetista do sistema. Atualmente, o conjunto de funções disponíveis inclui as funções trigonométricas mais comuns (sin, cos, tan, asin, acos, atan, etc.), e a função logarítmica (log). Também estão disponíveis as constantes PI (pi) e *épsilon* (e).

Tabela 5.1 - Conjunto de operadores disponíveis

Operador	Descrição
+	Adição
-	Subtração ou menos unário
*	Multiplicação
/	Divisão
==	Igual
!=	Diferente
>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual
&&	E lógico
||	OU lógico
!	Negação lógica

Como exemplo, vamos supor uma expressão que busque avaliar as condições de três informações sobre uma criança em um berço dotado de sensores especiais: a temperatura (variável ***t***), o batimento cardíaco (variável ***bc***) e a freqüência respiratória (variável ***fr***). Uma expressão que após avaliada logicamente com resultado verdadeiro, indicaria uma situação de alerta (ou emergência), poderia ser:

```
"t > 37.5 && (bc >= 140 || bc <= 80) && fr > 20"
```

Figura 5.14 - Expressão hipotética sobre temperatura, batimento cardíaco e freq. respiratória

Classe abstrata Alerta

A classe abstrata **Alerta** (fig. 5.15) fornece a estrutura e o comportamento padrão para objetos que têm por finalidade enviar uma mensagem de alerta a um usuário. Atualmente, para o envio de mensagens através do serviço SMS, utiliza-se o protocolo SMTP (*Simple Mail Transfer Protocol*), o que resultou na construção da classe

SMSbyMail. Esta classe faz uso de um objeto da classe *SendMail* (v. Anexo 1), oriunda do pacote *com.ringlord.smtp*.

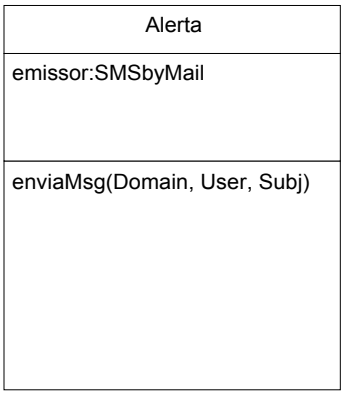

Figura 5.15 - A classe Alerta

Classe abstrata Atuador

A classe abstrata **Atuador** fornece o comportamento padrão e a infraestrutura de dados para que sejam especializadas classes que têm por finalidade modificar de alguma forma o estado do ambiente sendo monitorado.

Mantém informações sobre onde e como alterar características dos objetos que estão sendo monitorados, entre as quais as características de cada objeto que podem ser modificadas e qual o domínio de valores aceito; também define o vínculo com o dispositivo físico responsável pela ação interventora.

Classe Registro e Interface IRegistro

A classe **Registro** (fig. 5.16) implementa toda a funcionalidade para efetuar e recuperar registros de atividades ("log"), o que permite realizar a depuração sobre o funcionamento do sistema, além da possibilidade de efetuar auditorias e recuperações históricas sobre o comportamento do ambiente monitorado. Entre seus principais métodos públicos figuram:

- **gravaEvento(Evento)** - grava um objeto Evento em disco, no arquivo apropriado;
- **recuperaEventos(Usuario, DataInicial, DataFinal)** - recupera uma lista de objetos Evento do arquivo apropriado tendo por critério a identificação do usuário e o intervalo de datas;

- **recuperaEventos(Monitor, DataInicial, DataFinal)** - recupera uma lista de objetos Evento do arquivo apropriado tendo por critério a identificação do monitor e o intervalo de datas;

Figura 5.16 - A classe Registro

A interface **IRegistro** (fig. 5.17) tem por finalidade estabelecer o vínculo entre a classe **Registro** e todas as outras classes que necessitem implementar recursos de gravação e recuperação de registro de atividades ("log"). Para isso, estabelece o compromisso de que suas classes implementadoras definam concretamente o método **registraEvento()**.

Figura 5.17 - A interface IRegistro

5.2.4 Recursos Utilizados na Construção do *Framework*

A implementação do *framework* VIGIA foi baseada na linguagem e no ambiente Java. Tal escolha baseou-se na crescente popularidade que o Java vem obtendo,

por tratar-se de uma linguagem multiplataforma, baseada no princípio de máquina virtual. Esse ambiente também oferece uma variedade grande e crescente de bibliotecas de classes (pacotes, na terminologia Java), o que favorece muito o processo de desenvolvimento. [SUN 9?, COR 97]

Outro aspecto que confere muita flexibilidade a sistemas escritos em Java é o fato de que a maioria das classes encontradas na rede Internet estarem disponíveis com o código fonte. Esse foi o caso do pacote *nFunk.parser* (v. Anexo 2), o qual permitiu que o mesmo fosse estendido, conforme já citado (classe **Gatilho**, na seção 5.2.2).

A fig. 5.18 abaixo mostra as relações existentes entre as classes projetadas para o *framework* VIGIA e as classes provenientes de outras fontes. Na parte de cima, figuram as classes do pacote Java padrão da Sun Microsystems. E, na parte inferior, dois outros pacotes obtidos por pesquisa na Internet, parcialmente apresentados nos anexos.

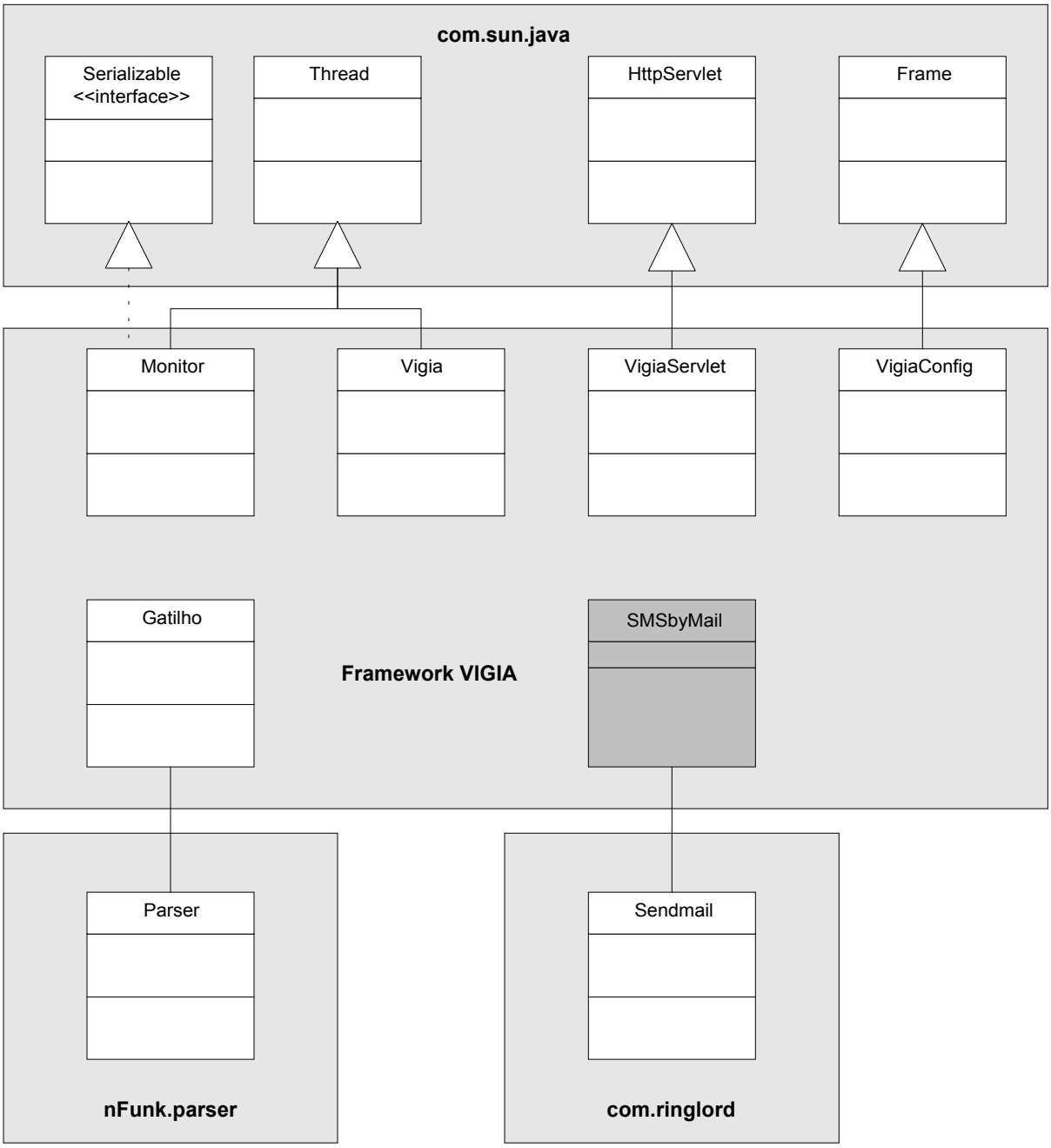

Figura 5.18 - Relações entre as classes do *framework* VIGIA e as classes externas

6 Conclusões

A percepção de uma demanda inexplorada, a possibilidade de criar algo novo, bem como a oportunidade de sedimentar com a prática os ensinamentos adquiridos durante o curso, além de explorar novos conhecimentos, foram as principais motivações para a elaboração do presente trabalho. Sua execução oportunizou o exercício dessas atividades de forma muito gratificante e produtiva.

O projeto do *framework* VIGIA atingiu um estágio bem avançado, embora se perceba que poderia ser melhorado. Ele fornece a infraestrutura básica e implementa o comportamento comum necessário a uma gama muito grande de aplicações voltadas à vigilância de ambientes, ampliando significativamente suas funcionalidades por permitir também a intervenção remota. A elaboração deste projeto envolveu a aplicação de uma série de técnicas e metodologias bem atuais, especialmente no que se refere à análise e projeto orientados a objetos.

A partir de uma especificação textual sobre o comportamento genérico de um sistema de vigilância com as características pretendidas, foram identificados os principais componentes de um sistema desse tipo, que quando mapeados do domínio da aplicação para o domínio computacional, resultaram nas principais classes que constituem o *framework*.

A implementação e experimentação de alguns de seus principais métodos ajudaram na construção e melhor definição do comportamento dessas classes, resultando num modelo comportamental completamente ajustado às expectativas.

As principais contribuições resultantes deste trabalho são as seguintes:

1. Apresenta um estudo sobre o estágio atual de desenvolvimento das tecnologias adotadas na comunicação de dados sem fio (capítulo 2);
2. Descreve os princípios básicos do serviço de mensagens curtas (SMS), seus componentes e características. Tal estudo reveste-se de importância significativa na atualidade, por ser este um serviço em ampla e crescente utilização (capítulo 3);
3. Aborda a emergente tecnologia WAP, que traz a Internet ao dispositivo móvel; neste estudo são apresentadas suas camadas de protocolos, e as linguagens WML e WML*Script*, utilizadas para descrever e construir os documentos enviados aos *handsets* (capítulo 4);

4. Fornece uma descrição metodológica (UML) do *framework* VIGIA, descrevendo seu comportamento, modelo arquitetônico e recursos fornecidos para a construção de sistemas genéricos de monitoração, alerta e intervenção remotos (capítulo 5).

Embora o *framework* projetado implemente o comportamento básico para os sistemas alvo, algumas melhorias podem ser feitas.

Por exemplo, os aspectos de segurança e tolerância a falhas, especialmente no que tange à disponibilidade do sistema, não foram integralmente tratados. Uma maneira imediata de monitorar o próprio funcionamento do sistema é definir um monitor que, a intervalos regulares, fique emitindo uma mensagem cuja única finalidade é comprovar o funcionamento do sistema. Outra forma mais sofisticada e confiável seria obtida com a interação de dois ou mais sistemas desse tipo, onde um monitoraria o outro.

Além disso, outros tipos de recursos para o envio das mensagens, como WAP *Push*, permitiriam o envio de informações mais detalhadas sobre o ambiente monitorado, podendo inclusive enviar imagens.

Anexo 1 – Classe Sendmail do pacote com.ringlord.smtp

```
package com.ringlord.smtp;

import java.io.*;
import java.net.*;
import java.util.Date;
import java.util.Enumeration;
import java.util.Properties;

/**
 *  <p>A  simple  sendmail  program/class  that  allows  an  application  to
assemble an email message
 * and then transmit it via a server's smtpd. The message must have a
sender, recipient, and
 * a message body; a subject line is optional.</p>
 *
 * <p>No attempt is made to provide for multiple recipients or any other
form of transfers
 * beyond the absolutely bare minimum.</p>
 *
 * @author  Udo Schuermann
 * @version 1.0 (24.3.1998)
 */
public class Sendmail
{
  private static StringBuffer responses = null;
  private String fromAddress;
  private String toAddress;
  private String subject;
  private String messageBody;
  private Properties misc = new Properties();
  ...
  ...
  /**
   * <p>Constructor builds a complete message.
        The fromAddress and toAddress must not be null.
   *    If the addresses do not contain a space,
        then they are converted to the format &lt;...&gt;
   *    The subject may be null, the body must not be null.</p>
   */
  public Sendmail( String fromAddress, String toAddress, String subject,
                   String messageBody )
  {
   setFromAddress(fromAddress);
   setToAddress(toAddress);
   setSubject(subject);
   setMessageBody(messageBody);
  }
  ...
  ...
  public static void main( String args[] )
  {
   if( args.length >= 3 ) {
     Sendmail msg;
     int messageFileIndex = 2;

     String subject = args[2];
     if( subject.startsWith( "-s" ) ) {
```

```
        if( subject.length() == 2 ) {
          subject = args[3];
          messageFileIndex = 4;
        }
      else {
        subject = subject.substring(2);
        messageFileIndex = 3;
      }
    }

    responses = new StringBuffer();
    String m = "";
    String s;
    try {
       BufferedReader i =
                   new        BufferedReader(new        InputStreamReader(new
                   FileInputStream(args[messageFileIndex])));
       while( (s = i.readLine()) != null )
         m = m + s + "\r\n";
       i.close();

       msg = new Sendmail( args[0], args[1], subject, m );
       String result = msg.sendMessage( "localhost" );
      }
    catch( Exception x ) {
        System.out.println( "Cannot read message from input file!" );
    }
  } else {
    System.out.println( "Usage: sendmail from to [ -s subject ] messagefile"
);
  }
}
...
...
public String sendMessage(String smtpdAddress, int port)
              throws IOException, Exception {
    Socket s;
    OutputStreamWriter o;
    BufferedReader i;
    if (isComplete()) {
       s = new Socket(smtpdAddress, port);
       o = new OutputStreamWriter(s.getOutputStream());
       i = new BufferedReader(new InputStreamReader(s.getInputStream()));
       sendLine(i, o, "HELO ringlord-sendmail", true);
       sendLine(i, o, "MAIL From:" + addressOf(fromAddress), true);
       sendLine(i, o, "RCPT To:" + addressOf(toAddress), true);
       sendLine(i, o, "DATA", true); // true ==> wait for response
       sendLine(i, o, "From: " + fromAddress, false);
       sendLine(i, o, "To: " + toAddress, false);
       if (subject != null)
          sendLine(i, o, "Subject: " + subject, false);
       sendLine(i, o, "Date: " + new Date(), false);
       boolean sendXMailer = true;
       Enumeration e = misc.keys();
       while (e.hasMoreElements()) {
          String key = (String) e.nextElement();
          sendLine(i, o, key + ": " + (String) misc.get(key), false);
          if (key.equalsIgnoreCase("X-Mailer"))
             sendXMailer = false;
       }
```

```
      if (sendXMailer)
         sendLine(i, o, "X-Mailer: RingLord Software Systems Atomic Java
Mailer v1.0", false);
      sendLine(i, o, "", false);
      sendLine(i, o, messageBody, false);
      sendLine(i, o, "", false);
      sendLine(i, o, ".", true);
      sendLine(i, o, "QUIT", true);
      try {
         o.close();
         i.close();
         s.close();
      } catch (Exception y) {}
      if (responses == null)
         return null;
      else
         return responses.toString();
   } else
      throw new Exception("Incomplete message");
}
...
...
  /**
   * <p>The default constructor</p>
   */
  public Sendmail()
  {
   this( null, null, null );
  }
}
```

Anexo 2 – Excertos do pacote nFunk.parser

```
package nFunk.parser;

import nFunk.parser.postfixMathCommand.*;
import nFunk.parser.exceptions.*;
import nFunk.parser.node.*;
import java.util.*;

/** class Parser
* Recursive decent parser for mathematical expressions.
* Uses the scanner class to recognize tokens.
* Creates a tree of the parsed expression out of nodes from the parser.node
package
*
* This class was extended to parse logical expressions too.
* It was made in November/2000 by Maurício Porto with the
*        support of Adriano Rochedo.
*/
public class Parser implements SharedConstants
{
   /** symTab: Symbol Table */
   private SymbolTable symTab;

   /** funTab: Function Table*/
   private FunctionTable funTab;

   /** scanner: Scanner */
   private Scanner scanner;

   /** status: Current status of the parser (stOk, stQuit, stError)*/
   private int status;

   /** outputString: added brackets to the expression for debugging*/
   private String outputString;

   /** tree: stores*/
   private Node tree;

/**
*  ExprList
*  <ExprList> ::= <level4> | <level4>','<ExprList>
*  List of Expressions seperated by a comma. Used for function arguments.
*   Returns a Stack of Nodes, with the top Node being the first expression
in the list
*
*/
   private Stack ExprList() throws InvalidExpressionException
   {
      Stack returnStack = null;
      Node topNode = level4(); //Accept the first expression from the list
      int token = scanner.Token();

      if (token == tCom)
      {// accept the comma and get the Stack representing the rest of the
list
         outputString += " , ";
         scanner.Accept(); //get next
```

```
         returnStack = ExprList();
      }
      else
      {// last element of the list, so create a new Stack
         returnStack = new Stack();
      }
      returnStack.push(topNode);
      return returnStack;
   }

/**
*  level0
*  <level0> ::= '('<level3>')' | <Number> | <Indentifier> |
*               <Identifier>'('<level3>')' | '-'<level1> //unary minus
*  Lowest level component of the expression. Can be Expression in brackets,
Number,
*  Identifier, a Function or an expression with a unary minus
*/
   private Node level0() throws InvalidExpressionException
   {
      int  token  =  scanner.Token();  //  holds  the  current  token  being
processed
      Node curNode;

      if (token == tLParen)
      {
         scanner.Accept(); // Accept '('
         curNode = level4();
         if (scanner.Token() != tRParen)
         {//couldn't find right Parenthesis
            throw new InvalidExpressionException("Expected ')'");
         }
         scanner.Accept(); // Accept ')'
      }
      else if (token == tNumber)
      {
         outputString += scanner.Number();
         curNode = new NumberNode(scanner.Number());
         scanner.Accept();
      }
      else if (token == tIdent)
      {
         outputString += scanner.SymbolName();
         String symbolName = scanner.SymbolName();
         //add error if Symbol is not found in the symboltable
         scanner.Accept(); //accept identifier
         if (scanner.Token() == tLParen) //function call
         {
            scanner.Accept();// accept '('

            Stack  parameterStack  =  ExprList();  //  get  the  parameter
expression

            if (parameterStack != null)
            {// good parameter stack, make function node
               if (funTab.containsKey(symbolName)){
/* function is in function table so create function node with stack of
parameters given by ExprList */
```

```
                  if                                                    (
((PostfixMathCommand)funTab.get(symbolName)).getNumberOfParameters()      !=
parameterStack.size())
                  {
                     throw                                            new
InvalidExpressionException(((PostfixMathCommand)funTab.get(symbolName)).get
NumberOfParameters() + " parameter(s) expected, " + parameterStack.size() +
" received");
                  }

                  curNode        =        new        FunNode(parameterStack,
(PostfixMathCommandI)funTab.get(symbolName));
               }
               else
               {//function not found in function table
                  curNode = null;
                  throw new InvalidExpressionException("Unkown Function: "
+ symbolName);
               }
            }
            else
            {// parameter tree was null
               curNode = null;
               throw new InvalidExpressionException("Missing parameters for
" + symbolName);
            }

            if (scanner.Token() == tRParen)
            {
               scanner.Accept(); //accept ')'
            }
            else
            {
               throw new InvalidExpressionException("Expected ')'");
            }
         }
         else //no '(', so variable
         {
            curNode = new VarNode(symbolName, symTab);
            if (symTab.containsKey(symbolName))
            {//recognized

            }
            else
            {//unknown identifier, assume value=1
               throw new InvalidExpressionException("Unknown Identifier");
            }
         }
      }
      else if (token == tMinus) {
         outputString += " -";
         scanner.Accept();

         Stack parameterStack = new Stack();
         parameterStack.push(level1());

         curNode = new FunNode(parameterStack, new UMinus());
      }
      else if (token == tNOT) {
         outputString += " !";
```

```
         scanner.Accept();

         Stack parameterStack = new Stack();
         parameterStack.push(level1());

         curNode = new FunNode(parameterStack, new NOT());

      }
      else if (token == tEnd) {
         scanner.Accept();
         curNode = null;
         throw new InvalidExpressionException("Unexpected end");
      }
      else {
         scanner.Accept();
         curNode = null;
         throw new InvalidExpressionException("Invalid Character");
      }

      return curNode;
   }
...
...
/**
*  Parser
*  Initializes fields
*/
   public Parser(Scanner scanner_in, SymbolTable symTab_in, FunctionTable
funTab_in) {
      status = stOk;
      scanner = scanner_in;
      symTab = symTab_in;
      funTab = funTab_in;
      outputString = "";
      tree = null;
    }
/**
*  Parse
*  Starts the parsing and sets tree equal to the top of the created tree
*/
   public void Parse() {
      try {
         tree = level4();
         if (tEnd != scanner.Token())
            throw new InvalidExpressionException("End not reached");
      }
      catch (InvalidExpressionException iee) {
         if (debugFlag) System.out.println(iee);
         status = stError;
         tree = null;
      }
   }
...
}
```

Glossário

Bearer Tipo de portadora, incluindo a técnica de controle de acesso múltiplo ao meio de comunicação.

Broadcast Difusão de sinal não endereçada, podendo ser recebido por qualquer um; como por exemplo, os canais abertos da TV e o rádio.

Browser Programa originalmente usado para navegação na *World Wide Web*.

Bytecode Código de instruções geralmente utilizados por máquinas virtuais.

CGI Scripting Seqüência de instruções em linguagem de alto nível para uso através de um servidor *Web*.

Datagrama Tipo de pacote de dados utilizado em serviços não orientados à conexão.

Denial-of-Service Impedimento de serviço. Técnica que busca sobrecarregar um servidor para tirá-lo do ar.

Download Quando um usuário copia para o seu computador um arquivo de outro computador, via modem ou rede, ele está fazendo um *download*. O termo *download* não encontrou ainda uma tradução apropriada em português. Em vez de *download*, algumas pessoas usam "baixar", "puxar" ou "descarregar".

e-mail É o correio eletrônico. Sistema de transmissão de mensagens eletrônicas.

Full-duplex Forma de comunicação que permite os dois sentidos simultaneamente. (telefone, por exemplo).

Gateway Interface entre duas redes distintas, em geral, de tipos diferentes.

Handset Dispositivo portátil pequeno de comunicação sem fio.

Homebanking Facilidade de acesso a serviços bancários através da Internet.

Hyperlink/Hiperlink Vínculo que pode ser percorrido entre dois documentos distintos ou partes de um mesmo documento. Também conhecido simplesmente por "link". Não há tradução definida em português, embora algumas pessoas usem "elo", "ponteiro" ou "vínculo".

JavaScript Java*Script* é uma linguagem de *scripting* baseada na linguagem Java, sendo, em geral, interpretados por *browsers*.

m-commerce O chamado comércio eletrônico “móbile” consiste basicamente na venda de produtos via Internet e utilizando um dispositivo móvel. Diversos tipos de mercadorias já são vendidos amplamente na rede mundial, como livros, flores, CDs e computadores.

Parser Decompositor e analisador de expressões gramaticais.

Proxy Agente intermediário que se encarrega de redirecionar a requisição de serviços ao verdadeiro servidor, atuando em nome deste e dessa forma, facilitando a requisição ao cliente.

Script Seqüência de instruções em linguagem de alto nível.

Simplex Forma de comunicação que permite apenas um sentido. (rádio, por exemplo).

Socket Mecanismo terminal para a comunicação entre processos (IPC).

Stub Pedaço de código que traduz a interface de uma função de e para uma forma canônica capaz de trafegar através de uma rede de comunicação.

tag Marcador utilizado para denotar o tipo de informação que o segue.

Referências Bibliográficas

[3GM 00] 3GMASTER. **3G - Terceira Geração Wireless**. Disponível por WWW em http://www.3gmaster.com.br/tutorial/3g/ (20/10/2000).

[ANY 00] ANYWHEREYOUGO.COM. **i-Mode**: Introduction to i-Mode - AnywhereYouGo.com. Disponível por WWW em http://www.anywhereyougo.com/ayg/ayg/imode/ (29/11/2000).

[BER 98] BERNERS-LEE, T. et al. **Uniform Resource Identifiers (URI)**: Generic Syntax. Disponível por WWW em http://www.ietf.org/rfc/rfc2396.txt (05/08/2000)

[BRA 98] BRAY, T. et al. **Extensible Markup Language (XML) 1.0 (Second Edition)**. Disponível por WWW em http://www.w3.org/TR/REC-xml (20/08/2000).

[BRA 98a] BRAY, T. et al. **Extensible Markup Language (XML) 1.0**. Disponível por WWW em http://www.w3.org/TR/1998/REC-xml-19980210 (02/08/2000).

[COR 97] CORNELL, G. e Horstmann, C.S. **Core Java**. São Paulo: Makron Books, 1997. 808 pp.

[DIA 00] DIAS, Adilson de Souza. **WAP – Wireless Application Protocol**: A Internet sem fios. Rio de Janeiro: Editora Ciência Moderna Ltda., 2000. 252p.

[FAR 99] FARLEY, Tom et al. **Cellular Telephone Basics by Tom Farley**: http://www.privateline.com. Disponível por WWW em http://www.privateline.com/Cellbasics/Cellbasics.html (10/07/2000).

[FIE 99] FIELDING, R. et al. **Hypertext Transport Protocol** – HTTP/1.1. Disponível por WWW em http://www.ietf.org/rfc/rfc2616.txt (05/08/2000).

[FOW 00] FOWLER, M; SCOTT, K. **UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos**. Porto Alegre: Bookman, 2000. 169p.

[GSM 00] GSM ASSOCIATION. **GSM – SMS Overview**. Disponível por WWW em http://www.gsmworld.com/technology/sms.html (0208/2000).

[IEC 00] INTERNATIONAL ENGINEERING CONSORTIUM. **Web ProForum Tutorial**: WAP. Disponível por WWW em http://www.iec.org/tutorials/wap/ (20/10/2000).

[IEC 00a] INTERNATIONAL ENGINEERING CONSORTIUM. **Web ProForum Tutorial**: Wireless SMS. Disponível por WWW em http://www.iec.org/tutorials/wire_sms/ (10/07/2000).

[IEC 00b] INTERNATIONAL ENGINEERING CONSORTIUM. **Web ProForum Tutorial**: SS7 Gateway Solution for Internet Access. Disponível por WWW em http://www.iec.org/tutorials/ss7_gate/ (13/09/2000).

[IEC 00c] INTERNATIONAL ENGINEERING CONSORTIUM. **Web ProForum Tutorial**: GSM. Disponível por WWW em http://www.iec.org/tutorials/gsm/ (27/10/2000).

[KIN 97] KING, P. et al. **Handheld Device Markup Language Specification**. Disponível por WWW em http://w3c.bilkent.edu.tr/TR/NOTE-Submission-HDML-spec.html (02/08/2000).

[McG 00] McGREGOR, Doug. **3G Wireless**. Disponível por WWW em http://www.wtc2000.org/programme/tu_02as.pdf (07/11/2000).

[POS 80] POSTEL, J. **RFC 768:** UDP. Disponível por WWW em http://www.pku.edu.cn/academic/research/computer-

center/tc/html/RFC768.html (01/07/2000)

[RAG 97] RAGGETT, D. et al. **HTML 4.01 Specification**. Disponível por WWW em http://www.w3.org/TR/REC-html40 (10/07/2000).

[REG 00] REGGIANI, Lucia. A vida pelos bytes. **Info Exame**. São Paulo, v. 15, n.175, p. 48-58. Out. 2000.

[RIB 00] RIBEIRO, Fabrício Jorge Lopes. Disponível por WWW em http://www.gta.ufrj.br/~fabricio/wap.htm (10/07/2000).

[SCI 98] THE SCIENCE AND TECHNOLOGY POLICY PROGRAM. **The Cellular Telephone**. Disponível por WWW em http://www.sri.com/policy/stp/techin2/chp4.html (10/07/2000).

[SEC 00] SECULUS INFORMÁTICA E TECNOLOGIA. **O que é WAP**. Disponível por WWW em http://www.wapon.com.br/wap.asp (04/11/2000).

[SUN 9?] SUN Microsystems. **The Java Tutorial**. Disponível por WWW em http://java.sun.com/docs/books/tutorial/index.html (15/10/2000).

[TAN 97] TANENBAUM, Andrew S. **Redes de Computadores**. Rio de Janeiro: Campus, 1997. 923p.

[TEL 00] TELEFONAKTIEBOLAGET LM ERICSSON. **CDMA 2000 - Code Division Multiple Access 2000**. Disponível por WWW em http://www.ericsson.se/technology/CDMA%202000.shtml (08/11/2000).

[WAP 00] WAP FORUM. **Wireless Markup Language Specification**. Disponível por WWW em http://www.wapforum.org/ (10/07/2000).

[WAP 00a] WAP FORUM. **WAP 1.2 Specification Suíte**. Disponível por WWW em http://www.wapforum.org/ (10/07/2000).

[WAP 00b] WAP FORUM. **WMLScript Language Specification**. Disponível por WWW em http://www.wapforum.org/ (10/07/2000).

[WAP 98] WAP FORUM. **Wireless Application Protocol Architecture Specification**. Disponível por WWW em http://www.wapforum.org/ (02/08/2000).

[WAP 98a] WAP FORUM. **Wireless Application Environment Overview**. Disponível por WWW em http://www.wapforum.org/ (02/08/2000).

[WAP 98b] WAP FORUM. **Wireless Session Protocol**. Disponível por WWW em http://www.wapforum.org/ (02/08/2000).

[WAP 98c] WAP FORUM. **Wireless Transaction Protocol Specification**. Disponível por WWW em http://www.wapforum.org/ (02/08/2000).

[WAP 98d] WAP FORUM. **Wireless Transport Layer Security Protocol**. Disponível por WWW em http://www.wapforum.org/ (02/08/2000).

[WAP 98e] WAP FORUM. **Wireless Datagram Protocol Specification**. Disponível por WWW em http://www.wapforum.org/ (02/08/2000).

[WAP 99] WAP FORUM. **Push Architectural Overview**. Disponível por WWW em http://www.wapforum.org/ (02/08/2000).

[WAP 99a] WAP FORUM. **WMLScript Crypto Library**. Disponível por WWW em http://www.wapforum.org/ (08/11/2000).

[YUR 00] YURI, Flávia. Deu SMS!. **Info Exame**. São Paulo, v. 15, n.175, p. 66-67. Out. 2000.

Um *Framework* para a Construção de Sistemas de Monitoração, Alerta e Intervenção Remotos

por

Wagner Luís Cardozo Gomes de Freitas

Monografia apresentada aos Senhores:

Prof. Cláudio Vianna Villela

Prof. Gil Carlos Rodrigues Medeiros

Prof. Maurício Nunes Porto

Vista e permitida a impressão.
Pelotas, 02 de fevereiro de 2001.

Prof. Maurício Nunes Porto
Orientador