UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

REINER FRANTHESCO PEROZZO

FRAMEWORK PARA CONSTRUÇÃO DE SISTEMAS SUPERVISÓRIOS EM DISPOSITIVOS MÓVEIS

Porto Alegre
2007

REINER FRANTHESCO PEROZZO

FRAMEWORK PARA CONSTRUÇÃO DE SISTEMAS SUPERVISÓRIOS EM DISPOSITIVOS MÓVEIS

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Automação e Instrumentação Eletro-Eletrônica.

ORIENTADOR: Carlos Eduardo Pereira

Porto Alegre

2007

REINER FRANTHESCO PEROZZO

FRAMEWORK PARA CONSTRUÇÃO DE SISTEMAS SUPERVISÓRIOS EM DISPOSITIVOS MÓVEIS

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Carlos Eduardo Pereira, UFRGS

Doutor pela Universidade de Stuttgart – Stuttgart, Alemanha

Banca Examinadora:

Prof. Dr. Dennis Brandão, USP.

Doutor pela Universidade de São Paulo – São Paulo, Brasil

Prof. Dr. João César Netto, UFRGS.

Doutor pela Universidade Católica de Louvain – Louvain, Bélgica

Prof. Dr. Renato Ventura Bayan Henriques, UFRGS.

Doutor pela Universidade Federal de Minas Gerais - Belo Horizonte, Brasil

Prof. Dr. Walter Fetter Lages, UFRGS.

Doutor pelo Instituto Tecnológico de Aeronáutica – São José dos Campos, Brasil

Coordenador do PPGEE: _____

Prof. Dr. Marcelo Soares Lubaszewski

Porto Alegre, março de 2007.

DEDICATÓRIA

Dedico este trabalho ao médico oncologista Dr. Túlio Trevisan, por ter aberto as portas que me trouxeram de volta à vida.

AGRADECIMENTOS

Quero agradecer a Deus, por ter me dado uma segunda chance. Aceitarei a Sua nova convocação somente no próximo século!

Entre todas as pessoas que participaram da minha trajetória nos últimos dois anos e que contribuíram para realização deste trabalho, gostaria de agradecer:

Aos meus pais e ao meu irmão, por todo o empenho em vencer gigantescos obstáculos que surgiram nas mais adversas condições;

À Manuela, por estar sempre ao meu lado, em todos os momentos, superando a tudo e a todos em pró de um causa;

A todos os médicos, enfermeiros e funcionários do Hospital Santa Rita, em especial aos médicos Fabiano Hahn Souza, pelo profissionalismo, e Alessandra Bastian Francesconi, pelo sorriso constante seguido da frase "você está ótimo, vai para casa..."; aos enfermeiros Carlos, pelas brincadeiras que descontraíram as sessões de quimioterapia, e Fabi, pelas dezenas de injeções aplicadas com extrema cautela;

Ao Sistema Único de Saúde (SUS), pelo ótimo tratamento prestado;

Ao meu orientador, o professor Carlos Eduardo, pela oportunidade de trabalharmos juntos, pela paciência de repetir mil vezes um mesmo conceito até que eu compreendesse, por custear as minhas participações nos eventos científicos e, principalmente, por todo o apoio pessoal cedido no momento em que mais precisei. Muito obrigado!

Ao Programa de Pós-Graduação em Engenharia Elétrica, PPGEE, pela oportunidade de realização do trabalho e à CAPES, pela provisão da bolsa de mestrado;

Aos colegas do PPGEE, pela amizade;

À Janice, uma pessoa muito querida no departamento.

RESUMO

Sistemas supervisórios são sistemas computacionais que permitem a monitoração de informações de processos produtivos ou de dispositivos de automação e de plantas industriais. Neste trabalho é proposto um *framework* para construção de tais sistemas, direcionada a dispositivos móveis, tais como *Personal Digital Assistants* (PDAs), telefones celulares e computadores de bolso. O *framework* proposto define uma arquitetura que é implementada gerando sistemas supervisórios com três características principais: (i) as telas de supervisão são construídas com base em um modelo orientado a objetos que utiliza conceitos da área da aplicação, mapeando o mundo real para um modelo computacional; (ii) o projeto de sistemas supervisórios apresenta flexibilidade com relação aos recursos computacionais disponíveis, oferecendo opções para armazenamento local ou remoto de dados multimídia, com o objetivo de não sobrecarregar os dispositivos móveis onde são executados; (iii) os sistemas supervisórios são capazes de se adaptarem dinamicamente às variações na Qualidade de Serviços (QoS) oferecidos pela infra-estrutura de comunicação, ajustando as suas telas gráficas em função de uma especificação de requisitos definidos em tempo de projeto e do nível de QoS obtido na rede em tempo de execução. Duas ferramentas computacionais são propostas e desenvolvidas no âmbito deste trabalho: (i) a primeira responsável pelo ambiente de desenvolvimento dos projetos de sistemas supervisórios, que resulta na geração automática de código em linguagem Java, correspondente à aplicação de supervisão; (ii) a segunda é responsável pela adaptação de mensagens e comunicação de dados entre as aplicações de supervisão projetadas e outros sistemas supervisórios e de controle, disponíveis no mercado. Os conceitos propostos neste trabalho foram validados através de três estudos de caso descritos na presente dissertação.

Palavras-chaves: Sistemas Supervisórios. Adaptabilidade e Flexibilidade. Dispositivos Móveis. Qualidade de Serviço. Orientação a Objetos.

ABSTRACT

Supervisory systems are computational systems which allow information monitoring of production processes or automation and technical plant devices. This paper proposes a framework for building supervisory systems, targeted to mobile devices, such as Personal Digital Assistants (PDAs), cell phones and Pocket PCs. The proposed framework allows the development of supervisory systems with three main characteristics: (i) supervision screens, which graphically depict the technical plant, are built using an object oriented model that uses concepts of the application area, allowing a direct mapping of real world concepts, such as automation devices to a computational model; (ii) the supervisory systems project presents flexibility about the available computational resources, offering options for local or remote storing of multimedia and graphical data, with the purpose of not overloading the mobile devices where they are executed; (iii) the supervisory systems runtime environment can dynamically adapt to variations in the Quality of Services (QoS) offered by the communication infrastructure, adjusting their graphic displays by comparing required and offered QoS. Two computational tools are proposed and developed in this work: (i) the first one supports the development of supervisory systems and allows, from an object-oriented model of the application, an automatic code generation - in Java language - of the supervision application; (ii) the second one is responsible for the online adaptation of messages and data communication among the projected supervision applications and other control and supervisory systems. The proposed concepts are validated through three case studies described in the present dissertation.

Keywords: Supervisory Systems. Adaptability and Flexibility. Mobile Devices. Quality of Service. Orientation to Objects.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ATM: Asynchronous Transfer Mode

CAD: Computer Aided Design

CE: Compact Edition

CMTP: Continuous Media Transport Protocol

DAO: Data Access Objects

DDE: Dynamic Data Exchange

DPE: Distributed Processing Environment

GCAR: Grupo de Controle, Automação e Robótica

HMI: Human-Machine Interface

HSNET: HomeSystems Network

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

I/O: Input/Output

IP: Internet Protocol

J2ME: Java 2 Micro Edition

JDK: Java Development Kit

JVM: Java Virtual Machine

LAN: Local Area Network

ODBC: Open Data Base Connectivity

OMG: Object Management Group

OPC: OLE for Process Control

PC: Personal Computer

PDA: Personal Digital Assistant

PLC: Programmable Logic Controller

QoS: Quality of Service

RCAP: Real-Time Channel Administration Protocol

RTIP: Real-Time Internet Protocol

RTT: Round-Trip Time

SCADA: Supervisory Control And Data Acquisition

SQL: Structured Query Language

TAS: Terminals Adaptation System

TIC : Tecnologia da Informação e Comunicação

UML: Unified Modeling Language

URL: Universal Resource Locator

WLAN: Wireless Local Area Network

WML: Wireless Markup Language

WTK: Wireless ToolKit

XMI: XML Metadata Interchange

XML: eXtensible Markup Language

XSL: eXtensible Stylesheet Language

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Avanços nas áreas de tecnologias de informação e comunicações (TIC) têm causado impactos significativos nas áreas de automação industrial e automação predial/residencial. Nessas áreas pode ser encontrado um conjunto de soluções, no qual destacam-se os sistemas supervisórios, conhecidos também por sistemas *Supervisory Control And Data Acquisition* (SCADA), que permitem o monitoramento de informações de um processo produtivo ou instalação física, onde são efetuadas a aquisição, a manipulação e a apresentação aos usuários das variáveis de processo e de controle/atuação (SILVA & SALVADOR, 2005).

Os sistemas de supervisão utilizam tecnologias computacionais e comunicação de dados para monitorar e controlar processos, coletando dados em ambientes complexos, eventualmente remotos, e a respectiva apresentação de modo amigável ao operador, com recursos visuais bem elaborados (MORAES & CASTRUCCI, 2001), (PEROZZO & PEREIRA, 2006).

A demanda crescente por sistemas supervisórios e a Internet resultam, constantemente, no desenvolvimento de novas soluções para a área de supervisão de processos, tais como: os sistemas supervisórios baseados na *web* e a integração entre diferentes sistemas computacionais (SOFT BRASIL, 2007).

Em geral os sistemas supervisórios operam em dois modos distintos:

▪ Modo de Desenvolvimento: modo no qual são definidas as variáveis de processo e controle a serem monitoradas/manipuladas, criadas as telas gráficas, definidos os alarmes e realizadas as configurações.

▪ Modo *Runtime*: modo de apresentação de telas, animação de objetos criados no modo de desenvolvimento e operação integrada com *Programmable Logic Controller* (PLC) ou *Personal Computer* (PC) (MORAES & CASTRUCCI, 2001).

As variáveis envolvidas num determinado processo de supervisão podem ser, por exemplo: temperatura, nível e vazão, nas quais o sistema supervisório pode verificar condições de alarmes, identificadas quando o valor da variável ultrapassa uma faixa de condição pré-estabelecida, sendo possível armazenar os registros em banco de dados, ativar som, enviar mensagens, alternar cores. A escolha do sistema supervisório a ser adotado depende de fatores resultantes de uma análise relacionada entre custo, benefício e exigências de cada sistema (SILVEIRA & SANTOS, 1998).

A construção das telas de um sistema supervisório pode ser realizada através da modelagem orientada a objetos, que auxilia na visualização e na compreensão de uma aplicação do mundo real. Com a modelagem orientada a objetos o mundo é modelado em tipos e comportamentos de objetos, sendo possível gerar um sistema supervisório através desses modelos (TIBOLA, 2000). A essência do desenvolvimento baseado em objetos é a identificação e a organização de conceitos do domínio da aplicação (RUMBAUGH, 1994). Com a utilização de ferramentas para modelagem, baseadas no paradigma de orientação a objetos, são desenvolvidos modelos que representem de forma significativa a complexidade do mundo real, especialmente aplicações industriais. Essas ferramentas permitem a criação de classes e a definição de atributos para os componentes de processo e dispositivos de automação.

Neste trabalho é utilizado para modelagem a *Unified Modeling Language* (UML), por ter se consolidado como uma linguagem padrão para descrição de modelos orientados a objetos, sendo adotada internacionalmente pela indústria e pela engenharia de *software.* Ela auxilia na definição de características do *software*, tais como seus requisitos, seu

comportamento, sua estrutura lógica, seus processos e sua necessidade física em relação a equipamentos de *hardware*. (GUEDES, 2005). Existem muitas ferramentas UML disponíveis no mercado, tais como a Rational Rose (IBM, 2007), a ArgoUML (ARGO, 2006) e a Visual-Paradigm for UML (VP-UML, 2006).

O aumento da complexidade dos sistemas computacionais como, por exemplo, os sistemas supervisórios, faz com que a modelagem e, por conseqüência, o desenvolvimento de *software* resultem em procedimentos difíceis de serem implementados, sendo identificadas muitas falhas de projeto relacionadas à arquitetura do sistema (ANDRADE, 2006). Para que uma arquitetura possa ser utilizada corretamente num determinado domínio de aplicação são exigidos do projetista conhecimento do problema que está abordando e experiência no desenvolvimento de sistemas computacionais.

Desse modo, uma solução utilizada em desenvolvimento de *software* e seguida também por este trabalho são os *frameworks*, que podem ser definidos como uma biblioteca de classes, concretas e abstratas, com padrões de interação entre objetos (ROGERS, 1997). Também podem ser definidos como um conjunto de classes cooperantes para a construção de projetos reutilizáveis (GAMMA *et al*., 2000) ou, ainda, como hierarquias de classes genéricas para um domínio de aplicações (SHALLOWAY & TROTT, 2004).

Os *frameworks* implementam uma arquitetura que pode ser utilizada de modo a tornar o desenvolvimento de sistemas computacionais mais rápidos e produtivos, facilitando a captura de decisões de projetos que são comuns ao domínio de uma aplicação e permitindo uma redução do tempo gasto com o desenvolvimento de novas soluções. A principal contribuição de um *framework* é a definição de sua arquitetura projetada para suportar todas as aplicações do domínio de maneira flexível e extensível (GAMMA *et al.*, 2000).

A exigência de que os sistemas supervisórios sejam executados em diferentes plataformas computacionais e utilizem diferentes modos de comunicação e exibição de dados,

faz com que o desenvolvimento de um *framework* para construção de sistemas supervisórios que operem em dispositivos móveis não seja algo trivial de ser implementado, uma vez que existem vários fatores a serem abordados. Por exemplo, um problema com o qual os desenvolvedores de sistemas que utilizam a comunicação *web* se deparam é a proliferação de inúmeros *browsers*, diferenciados pela capacidade e pelo suporte a padrões atuais, formas e estilos de visualização das informações. Alguns *browsers* consideram somente a parte textual ignorando a questão gráfica, o que seria uma solução viável para usuários com conexão lenta à Internet. Outros *browsers* consideram a parte multimídia, mas a utilização seria recomendada apenas para usuários com conexão rápida.

Outro fator importante é que com o aumento significativo de dispositivos portáteis que acessam a Internet são apresentados alguns problemas, tanto no nível do usuário, quanto do programador. Esses novos dispositivos portáteis utilizam diferentes larguras de banda e suas capacidades de visualização são variadas (KAPLAN & LUNN, 2001).

Um problema adicional é que a maioria dos dispositivos portáteis suporta somente alguns formatos de imagens, as quais devem ser convertidas antes de serem visualizadas. Ainda, as grandes oscilações na largura de banda em redes sem fio (“*wireless”)* também devem ser consideradas na utilização de sistemas multimídias adaptáveis e nas comunicações móveis (FEI YU *et al.*, 2004).

Além disso, o crescente número de usuários que utilizam o serviço de comunicação móvel resulta, ainda, em um aumento significativo de acesso a dados multimídia em redes sem fio. A comunicação multimídia nessas redes requer bem mais do que dispositivo móvel e canal de acesso: é necessário considerar, por exemplo, o tamanho dos dados multimídia, o tráfego de rede, a capacidade de processamento do dispositivo móvel e outras características que incluem tamanho da tela, tipos de comunicação e forma de visualização (DONG-GI LEE & DEY, 2004). Como os dados multimídias presentes nos sistemas supervisórios podem

trafegar por diversas redes de comunicação de dados é utilizado o conceito de qualidade de serviço, mais comumente conhecido como QoS (do inglês "*Quality of Service*"), que se refere à maneira como um sistema desempenha seus serviços, estando normalmente relacionada à taxa útil de transmissão de dados, atraso e *jitter*. Ações de computação e comunicação na adaptação de QoS podem ser executadas para melhorar parâmetros de qualidade na aplicação e recursos de comunicação (BASTO & FREITAS, 2005).

Segundo (AURRECOECHEA *et al.*, 1998) para que as aplicações confiem na transferência dos dados multimídia é necessário que a QoS seja configurável e previsível em todo o sistema. Também é importante que todos os elementos fim-a-fim presentes na arquitetura dos sistemas distribuídos trabalhem em total colaboração e harmonia, para alcançar o comportamento esperado da aplicação (BORCOCI *et al.*, 2005). Além de uma negociação inicial, um sistema multimídia distribuído deve planejar o monitoramento e a renegociação. Uma solução seria disponibilizar o mesmo nível de QoS a todos os clientes, mas isso geralmente resulta em um número significativo de usuários que receberão um percentual de QoS insatisfatório (HESSELMAN *et al*., 2001).

Embora o conceito de QoS tenha sua origem na especificação dos níveis de serviços de redes, é estendido até o nível de usuário, para que defina um grau de qualidade nos serviços (YAMAZAKI, 2003). Em aplicações que utilizam dados multimídia, como é o caso dos sistemas supervisórios, é fundamental que, além do nível de QoS definido pelo usuário, a aplicação também se encarregue de adaptar o modo de visualização das informações, baseando-se no nível de QoS disponível pela infra-estrutura de comunicação.

1.2 MOTIVAÇÃO

Os dispositivos móveis estão em um nível tecnológico que inclui telas gráficas coloridas com boa resolução, acesso à Internet em alta velocidade, boa capacidade de

processamento local e um custo acessível. A possibilidade de integrar a supervisão de processos automatizados com a mobilidade da informação torna-se bastante atrativa, uma vez que há uma crescente demanda industrial pela supervisão móvel de ambientes industriais automatizados, sendo essa uma área em potencial explorada por grandes empresas na área de automação industrial e integração de sistemas computacionais.

A supervisão de processos através da computação móvel pode oferecer ao operador liberdade para se movimentar fisicamente pela indústria e, ainda assim, estar supervisionando um determinado processo através de seu dispositivo móvel. Poderia, também, estar fora dos limites físicos da indústria e supervisionar de forma remota os processos industriais.

A motivação também é fruto dos desafios científicos e tecnológicos a serem tratados na área da computação móvel, como as questões de adaptabilidade em função das variações de serviços presentes em redes sem fio, a flexibilidade com relação aos recursos de *hardwar*e dos dispositivos móveis, a portabilidade das aplicações de supervisão projetadas e o mapeamento de conceitos presentes no domínio do problema para os conceitos computacionais.

1.3 OBJETIVOS

O objetivo deste trabalho é a criação de um *framework* para ser utilizado no desenvolvimento de sistemas supervisórios que serão executados em dispositivos móveis. A partir da definição de uma arquitetura podem ser criados e reutilizados projetos de sistemas supervisórios, com geração automática de código para a plataforma alvo de supervisão. Dentre as funcionalidades a serem disponibilizadas pelo *framework*, destacam-se (i) a especificação de uma arquitetura genérica e que possa ser reusada em diferentes aplicações do domínio; (ii) o desenvolvimento das telas de supervisão e da estrutura da informação baseados em modelos orientados a objetos; (iii) a flexibilidade com os recursos

computacionais disponíveis; (iv) a adaptação dinâmica dos sistemas supervisórios projetados que consideram a qualidade dos serviços oferecidos pela infra-estrutura de comunicação como requisito para visualização dos dados; (v) a integração com outros sistemas supervisórios e a comunicação com controladores.

2 ANÁLISE DO ESTADO DA ARTE

Este capítulo tem por objetivo oferecer uma breve descrição do estado da arte sobre a construção de sistemas supervisórios e os requisitos necessários para transformar dispositivos móveis em plataformas de supervisão. Serão apresentadas algumas técnicas de adaptabilidade e flexibilidade, QoS em sistemas multimídia, utilização da orientação a objetos e algumas ferramentas para desenvolvimento de sistemas supervisórios.

2.1 SISTEMAS SUPERVISÓRIOS

Nesta seção serão analisados alguns sistemas supervisórios e suas principais funcionalidades, identificando como são consideradas as soluções para comunicação, acesso e visualização de dados, bem como a plataforma alvo para execução das aplicações de supervisão.

Uma das ferramentas de sistemas supervisórios escolhidas para análise é a Wizcon, por deter uma solução completa em sistemas de supervisão baseados em Internet e utilizados em diversos setores, tais como: automação industrial, saneamento e energia. A família de produtos dos sistemas Wizcon está consolidada no mercado há mais de dezoito anos, tendo ultrapassado a marca de cinqüenta mil licenças liberadas para utilização em todo o mundo. (ELUTIONS, 2006) (AXEDA,2007).

Outra ferramenta analisada é o Elipse SCADA, escolhido por possuir no seu conjunto de soluções, um ambiente de execução para sistemas supervisórios direcionado a dispositivos móveis e por ser uma ferramenta bastante conhecida e utilizada pelo grupo de pesquisa no qual este trabalho foi desenvolvido.

Por fim é analisado o Pocket GENESIS, escolhido por apresentar uma ferramenta de geração de sistemas supervisórios voltada, exclusivamente, para plataformas móveis. O

Pocket GENESIS é desenvolvido pela empresa Iconics, líder em automação industrial com utilização da *web* e sistemas inteligentes de manufatura (ICONICS-UDS, 2007).

2.1.1 Wizcon

A Wizcon (WIZCON, 2006) é uma solução SCADA para controle e informação que pode ser visualizada através de um navegador de Internet. As ferramentas de desenvolvimento da Wizcon permitem acesso a banco de dados de aplicações que contém informações de processos e de alarmes. A Wizcon permite a criação de *interfaces* com imagens, gráficos e resumos de evento. É possível executar os aplicativos nas estações de trabalho ou publicar as informações em um servidor *web*, através de utilitários de conversão *HyperText Markup Language* (HTML) e *applets* Java aliados a um conjunto de soluções complementares que permitem controlar e supervisionar o processo de produção remotamente, estando conectado à Internet ou à Intranet.

A Figura 2.1 apresenta uma visão geral da arquitetura de comunicação suportada pela Wizcon.

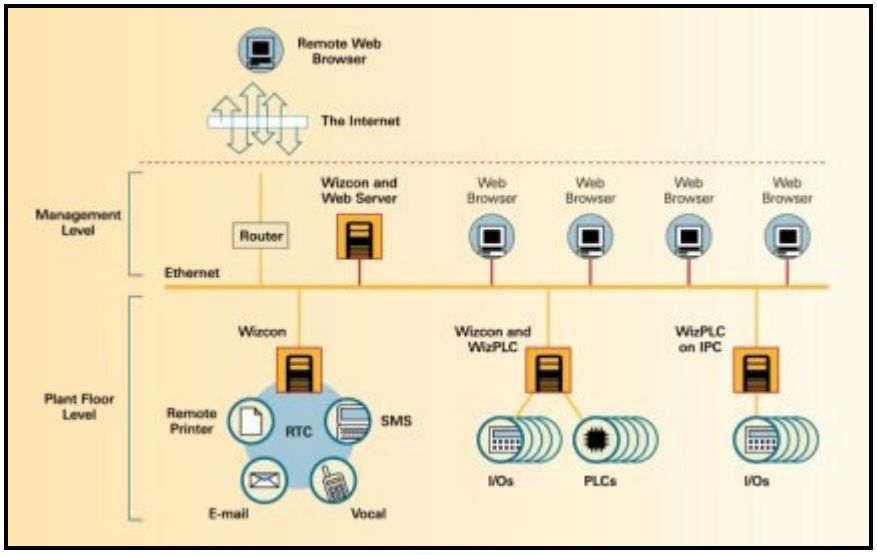

Figura 2.1 Arquitetura de Comunicação do Sistema Wizcon [WIZCON, 2006]

A Wizcon possui *drivers* para diversos PLCs e equipamentos de automação, com conectividade a banco de dados e sistemas corporativos. Também possui um módulo que

permite o planejamento, a programação e a execução de uma variedade de tarefas baseadas em especificações temporais. A interface calendário oferece um método para definir atividades a serem executadas periodicamente. É possível criar relatórios customizados em vários formatos, contando com um módulo, opcional, que permite converter as imagens criadas por ferramentas de *Computer Aided Design* (CAD) em imagens utilizadas nas aplicações SCADA.

Além da conversão de imagens CAD, a Wizcon possui um editor próprio de imagens que permite ao operador visualizar uma parte ou todo processo de controle. Os usuários pedem criar objetos do tipo Dinâmicos (as imagens são associadas com as *Tags*, onde cada alteração na *Tag* gera, conseqüentemente, uma alteração na visualização do objeto) e objetos do tipo Alarme (imagens são associadas com o alarme, de modo que este defina o comportamento da imagem).

Uma aplicação Wizcon é composta por uma série de arquivos que facilitam futuras alterações em aplicações já utilizadas. Também é possível criar macros: operações que podem ser executadas ao clicar um botão na tela da aplicação ou de acordo com uma situação pré-estabelecida e trocar informações entre aplicações que utilizam base de dados, através da interface *Open Data Base Connectivity* (ODBC) e via OLE *for Process Control*, (OPC).

Entretanto, a Wizcon não possui soluções para automação industrial e predial/residencial baseado em dispositivos móveis. Além disso, não são consideradas as questões de portabilidade, flexibilidade com relação ao *hardware* no qual é executado o sistema de supervisão e, adaptabilidade de acordo com a qualidade e disponibilidade dos serviços que são oferecidos pela infra-estrutura de comunicação de dados.

2.1.2 Elipse SCADA

O Elipse SCADA (ELIPSE, 2006) é um *software* de supervisão de processos que permite a monitoração de variáveis e o acionamento de equipamentos. O Elipse SCADA contém funções de monitoração, controle, importação de imagens, alarmes, autenticação de usuários, servidor e cliente *Dynamic Data Exchange* (DDE), podendo ser programado através da linguagem Elipse *Basic* e servidor de aplicações remotas.

Ademais, oferece suporte para aplicações que utilizam armazenamento de dados, tratamento de informações e geração de relatórios complexos, sendo disponibilizadas funções para históricos, relatórios, controle estatístico de processos e *log* de alarmes em disco. Permite, outrossim, que aplicações troquem informações com banco de dados, suportando comunicação ODBC e *Data Access Objects* (DAO).

O Elipse SCADA possui ferramentas adicionais que trabalham em conjunto com o *software* e que permitem acrescentar funcionalidades, como a monitoração de sistemas por imagens e a supervisão *web* de processos através de um navegador de Internet.

Somado a isso conta com a ferramenta chamada *Organizer*, a qual possibilita uma visão geral da aplicação através de um navegador hierárquico de funcionalidades, parecido com uma árvore de diretórios e subdiretórios, onde a aplicação é a raiz e os objetos são agrupados pelo seu tipo: *tags*, telas, alarmes e demais objetos. Ao selecionar qualquer ramo da aplicação é exibida, no lado direito da janela, toda a informação referente à propriedade do objeto. O *Organizer* permite realizar tarefas rapidamente como, por exemplo, a programação de *scripts* Elipse *Basic* em botões de comando.

A supervisão de processos no Elipse SCADA é efetuada através da leitura de variáveis de processos, associada a valores de variáveis com *tags* do sistema. Quando as *tags* são criadas o usuário pode organizá-las em grupos para facilitar a procura e identificação durante o processo de configuração.

Com o desenvolvimento das novas tecnologias de comunicação móvel, a empresa Elipse *Software* disponibilizou o Elipse SCADA *Compact Edition* (CE): uma plataforma que permite executar aplicações feitas no ambiente de desenvolvimento do Elipse SCADA em dispositivos baseados no sistema operacional Microsoft Windows CE, incluindo telefones celulares e PDAs.

Porém, o Elipse CE também não considera as questões de adaptação de visualização de acordo com requisitos de QoS pré-estabelecidos, de oscilações ocorridas nas redes sem fio e de flexibilidade no armazenamento de dados de acordo com a plataforma alvo. Outro fator importante é que a tela de supervisão projetada no Elipse SCADA para o dispositivo móvel assume a mesma resolução criada originalmente no PC, o que implica em rolagem de barras, verticais e horizontais, para visualização dos dados de supervisão no dispositivo móvel.

2.1.3 Pocket GENESIS

Pocket GENESIS (ICONICS, 2006), é um conjunto de ferramentas de automação com arquitetura para executar *Human-Machine Interface* (HMI) nos dispositivos baseados em plataformas *Pocket* PC com sistema operacional Windows CE.

Com o Pocket GENESIS é possível realizar visualização e interação com telas de HMI, permitindo conexão com PLCs, dispositivos de Input/Output (I/O) e barramentos industriais através de OPC. Ele gerencia alarmes e eventos, incluindo funcionalidades de filtros e opções variadas de visualizações.

Os dados de supervisão são representados graficamente, o que possibilita a análise e a integração com os sistemas de informações gerenciais da empresa. O Pocket GENESIS oferece suporte a comunicações em redes sem fio, atuando como um *gateway* entre as aplicações do Pocket GENESIS e uma estação de trabalho.

A Figura 2.2 apresenta um exemplo da comunicação dos dados, em que as ordens de trabalho são geradas e enviadas para o banco de dados da planta. A partir desse momento os operadores podem efetuar a supervisão dos processos ou a equipe responsável pela manutenção da planta pode realizar operações necessárias.

Figura 2.2 Exemplo de Comunicação de Dados com Pocket GENESIS [ICONICS, 2006]

A Figura 2.3 apresenta uma tela com a visão geral da supervisão de processos utilizando o Pocket GENESIS.

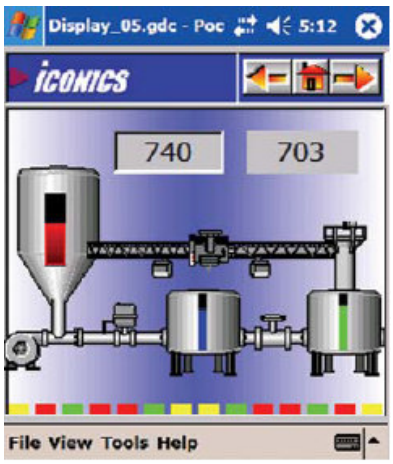

Figura 2.3 Supervisão de Processos com Pocket GENESIS [ICONICS, 2006]

O Pocket GENESIS permite, ainda, a sua integração com outras aplicações que utilizam banco de dados tais como: Microsoft SQL Server, Microsoft Access ou Oracle.

No Pocket GENESIS é considerada uma funcionalidade da comunicação móvel, sendo implementada para atuar com ou sem serviço de rede, por exemplo: caso um usuário esteja se movimentando por uma área inoperante da planta, o Pocket GENESIS possui um mecanismo que armazena os dados e os envia logo após detectar uma conexão de rede. Ou seja, mesmo assim essa funcionalidade é implementada considerando apenas se há ou não uma conexão de rede disponível no momento, não sendo verificados a QoS da rede, a degradação do sinal e uma possível adaptação de visualização em tempo de execução do sistema supervisório.

Por outro lado, o Pocket GENESIS possui todas as funcionalidades de uma HMI para *desktops*, conta com um ambiente de desenvolvimento para vários dispositivos alvos, sendo possível gerar aplicações de supervisão totalmente portáveis para plataformas baseados no sistema operacional Windows CE.

2.2 O USO DE MODELOS ORIENTADOS A OBJETOS NO DESENVOLVIMENTO DE SISTEMAS SUPERVISÓRIOS

A ferramenta *Supervisory Designer* foi desenvolvida no âmbito da dissertação de mestrado de Leandro Tibola (TIBOLA, 2000) visando integrar modelos orientados a objetos e sistemas supervisórios. O objetivo dessa integração entre uma ferramenta de modelagem e uma ferramenta de desenvolvimento de sistemas supervisórios é tornar o trabalho de projeto e implantação mais rápido e produtivo através da reutilização de componentes e projetos.

A ferramenta *Supervisory Designer* permite uma definição de características e valores em tempo de projeto, permitindo a vinculação de "*Tags*" de sistemas supervisórios (implementação realizada utilizando o sistema supervisório Elispe SCADA) com atributos e métodos de objetos modelados usando a ferramenta SIMOO-RT (BECKER & PEREIRA, 2002) que é um ambiente de modelagem, simulação e geração de código para aplicações tempo-real. Outra possibilidade é seleção do modo de visualização desejado, no qual o

usuário pode escolher, editar ou criar uma representação. A Figura 2.4 apresenta o esquema de integração entre as ferramentas *Supervisory Designer* e Elipse SCADA Windows.

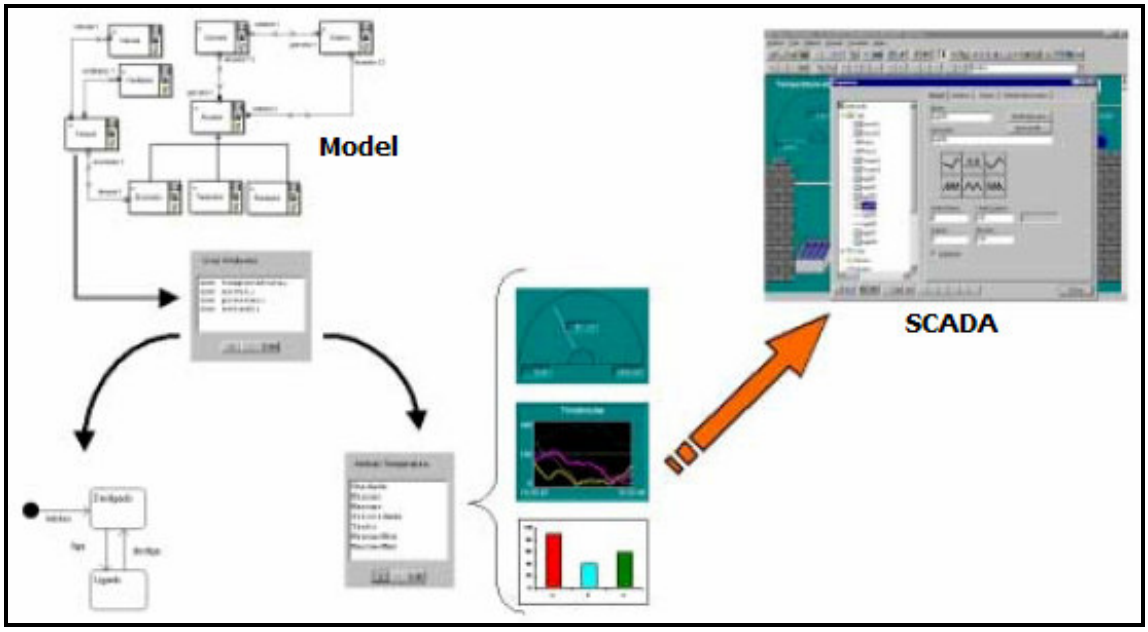

Figura 2.4 Esquema de Integração Supervisory Designer e Elipse SCADA [TIBOLA, 2000]

O acesso a estruturas de classes do SIMOO-RT permite a ligação entre atributos e elementos de visualização. A escolha pode ser feita entre um dos tipos de visualização disponíveis: *Gauge*, *Display*, *Bargraph* ou *Bitmap*. É criado, então, um objeto que tem como informações a classe atual, o nome da classe, os atributos da classe e os dados dos alarmes para os atributos da classe.

O ambiente é baseado em modelos criados no SIMOO-RT e a partir desses é gerada uma estrutura de dados compatível com a usada pelo programa Elipse SCADA Windows, tanto em nível de visualização e simulação, quanto do funcionamento na planta industrial.

2.3 ADAPTABILIDADE E FLEXIBILIDADE

Nesta seção serão analisadas propostas para adaptação e representação de informações, sendo consideradas as plataformas computacionais, as formas e as prioridades de visualização.

2.3.1 FlexXML

O FlexXML (KAPLAN & LUNN, 2001) é uma proposta de extensão para *eXtensible Markup Language/eXtensible Stylesheet Language* (XML/XSL) a fim de permitir a definição de *sites* que possam se adaptar às variações de parâmetros de comunicação, bem como possam ser exibidos em *browsers* de diversos dispositivos de *hardware* (PDAs, telefones celulares, PCs). Usando XML, um desenvolvedor *web* cria uma especificação simples com conteúdo para um *site* e variados estilos XSL, ocorrendo adaptação a diferentes tipos de largura de banda, *browsers* e dispositivos. Por exemplo, uma especificação XSL poderia ser usada na criação de uma apresentação HTML para uma conexão de alta velocidade, com uma exibição baseada em gráficos, utilizando o *Internet Explorer*. Outro estilo XSL, para a mesma especificação XML, poderia criar uma apresentação usando *Wireless Markup Language* (WML) para uma conexão de baixa velocidade, com uma exibição baseada em texto, utilizando um *browser* de telefone celular. Baseado no ambiente particular do *browser* de cada usuário que pretende acessar o *site*, o *framework* FlexXML seleciona automaticamente o estilo XSL, aplica este à especificação XML e transmite o conteúdo apropriado. Assim, os desenvolvedores *web* não necessitam manter múltiplas versões de um mesmo *site*.

Ao acessar um *site*, o conteúdo mais apropriado ao ambiente do *browser* é gerado automaticamente. O FlexXML também permite que o usuário customize seu ambiente. Os usuários podem indicar os tipos de documentos que eles esperam de um *site* ou definir suas conexões de rede atuais, sendo utilizada essa informação para exibir um conteúdo que melhor se adapte ao *browser* do usuário.

A Figura 2.5 apresenta uma visão conceitual do FlexXML e sua relação com os serviços w*eb*. O *framework* se comunica com os quatro seguintes componentes: um *proxy* de aplicação de *browser*, um servidor *Web proxy*, um *parser* XML e um processador XSL.

O *proxy* do *browser* é uma extensão transparente de diversos *browsers* existentes. Seu objetivo é comunicar características do *browser* diretamente a um servidor *web proxy*. As características de customização do *browser* incluem uma indicação de largura de banda atual e uma especificação de somente texto ou conteúdo de gráficos, enquanto as características do *browser* incluem o tipo de *browser* e seu dispositivo.

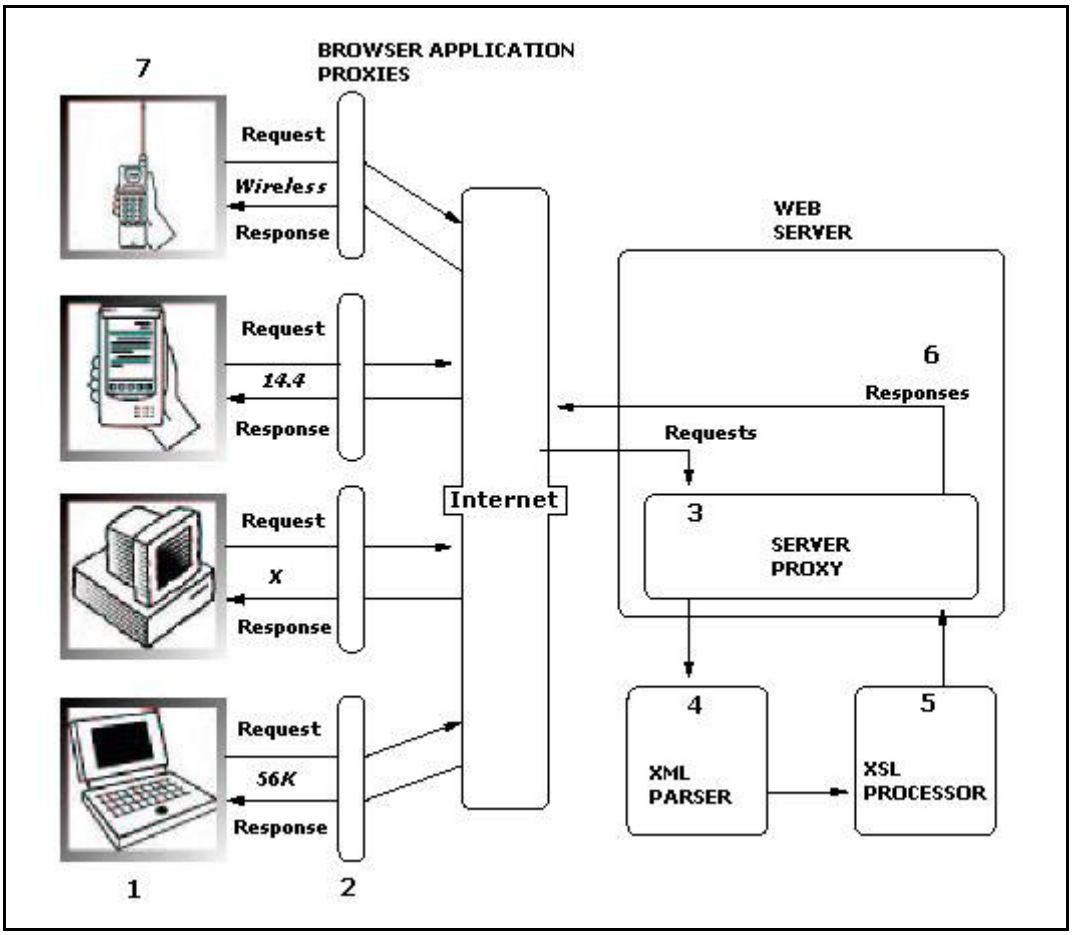

Figura 2.5 O Framework FlexXML [KAPLAN & LUNN, 2001]

Como ilustrado na Figura 2.5 o *proxy* do servidor *web* situa-se entre o *web browser* do usuário e um servidor *web*. Seu objetivo é interceptar e processar os cabeçalhos *HyperText Transfer Protocol* (HTTP) que são adicionados pelo *proxy* de aplicação do *browser*. O *proxy* do servidor *web* utiliza essa informação para selecionar um estilo XSL que melhor satisfaz as características do *browser* do usuário.

O *proxy* do servidor encaminha um documento XML para o XML *parser*, gerando uma representação de árvore do documento XML, enviando a árvore para o processador XSL. Usando o estilo XSL selecionado pelo *proxy*, o processador XSL percorre a árvore e gera a representação apropriada para o *web browser* seguindo as regras de tradução dadas no estilo XSL.

Por exemplo, um cabeçalho HTTP criado pelo FlexXML poderia indicar uma requisição para documentos de somente texto, enquanto outro cabeçalho poderia indicar uma solicitação para documentos gráficos. Usando essa informação, um *proxy* do servidor *web* seleciona automaticamente um estilo XSL que melhor se enquadra na transformação de um documento XSL para uma representação desejada.

2.3.2 @TAS

O @*Terminals Adaptation System* - @TAS (DI NITTO *et al.*, 2003) é uma infra-estrutura de acesso universal e adaptação de conteúdos *web*. A infra-estrutura atua como um mediador entre terminais de usuários, provedores de serviços e seleção de conteúdos. A Figura 2.6 apresenta os principais componentes funcionais da infra-estrutura @TAS, onde, o componente *Terminal Manager* realiza a comunicação física, reconhecendo os terminais e capturando o contexto de entrega correspondente. Já o componente *Session Manager* gerencia a interação entre os usuários e os serviços, rastreando as sessões de navegação. Esse componente também é responsável por gerenciar a troca de terminais durante a sessão de navegação, permitindo aos usuários suspender a sessão e retomá-la em um terminal diferente. O componente *Service Adapter* executa a ação de adaptação, selecionando os filtros disponíveis para adaptar o conteúdo ao contexto de entrega atual e o componente *Service Integrator* implementa uma interface pública que os provedores de serviços podem utilizar

para acessar informações atualizadas do contexto de entrega, da sessão de navegação e dos dados sobre os usuários.

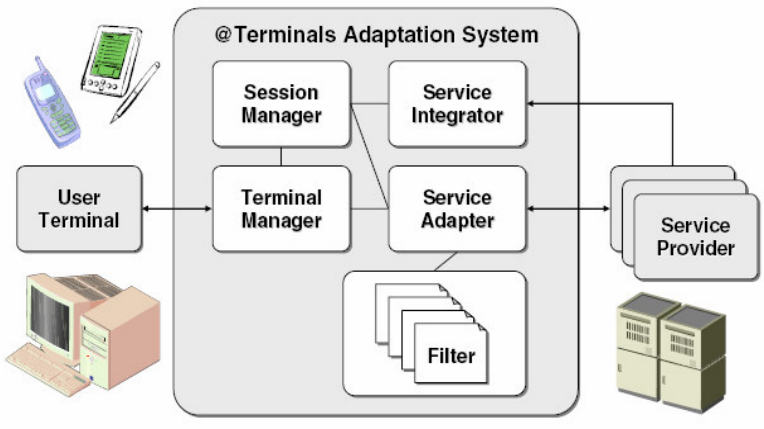

Figura 2.6 A Arquitetura @TAS [DI NITTO *et al.*, 2003]

A adaptação de conteúdo realizada pelo @TAS é dinâmica: a decisão sobre o modo e o conteúdo que deve ser adaptado para a entrega é obtida somente quando o conteúdo é solicitado, pois é apenas nesse momento que é identificado quem originou tal solicitação.

A infra-estrutura pode realizar adaptações genéricas em conteúdos básicos da *web* como, por exemplo, redimensionamento de imagem para possibilitar a sua exibição em vários tamanhos de telas e eliminação de um tipo específico de formatos não suportados.

A adaptação de conteúdo é o resultado da composição e execução de filtros. Os filtros são tratados pelo @TAS como uma caixa-preta que implementa uma interface descrita por seu perfil. O perfil dos filtros são descrições XML que especificam o tipo de transformação que o filtro pode realizar, o tipo de entrada e saída e as condições de ativação. A Figura 2.7 mostra o resultado de uma adaptação realizada em uma página *web*. No lado esquerdo da figura é visualizada a página *web* através do *browser* Microsoft Internet Explorer. Já no lado direito da figura é visualizada a página *web* no emulador *Nokia Mobile ToolKit*, representando como uma mesma página seria exibida num dispositivo móvel real. No emulador a imagem gráfica foi removida, uma vez que sua visualização foi considerada como de baixa prioridade para esse caso, sendo exibidas apenas as informações textuais.

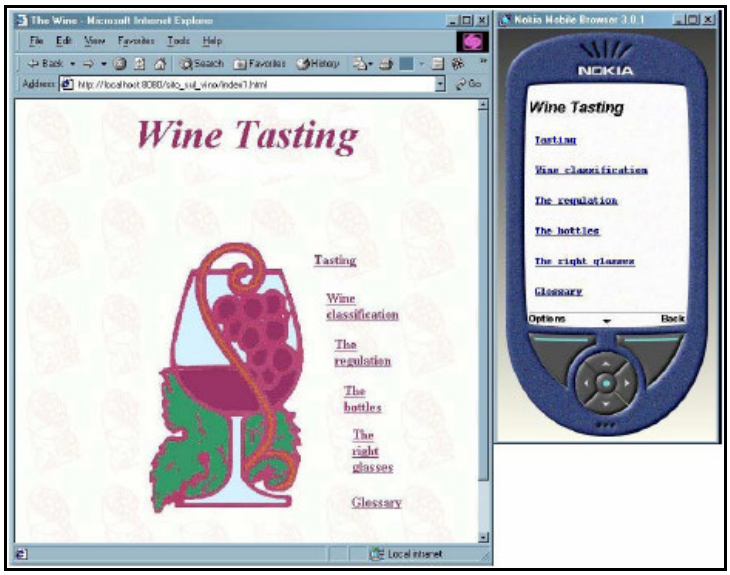

Figura 2.7 Resultado da adaptação realizada pelo @TAS [DI NITTO *et al.*, 2003]

2.4 QoS EM SISTEMAS MULTIMÍDIAS

De acordo com (AURRECOECHEA *et al.*, 1998), garantir QoS em sistemas multimídia distribuídos é essencialmente uma questão fim-a-fim, isto é, de aplicação à aplicação. Considerando, por exemplo, um dispositivo executando remotamente uma seqüência de áudio e vídeo: na plataforma de sistema distribuído, garantias de QoS deveriam se aplicar para o fluxo completo de mídia, no servidor remoto, na rede e nos pontos de entrega. Isso poderia exigir teste de admissão fim-a-fim e reserva de recurso no primeiro momento, seguido pelo controle de fluxo na rede e pelo monitoramento da QoS oferecida.

De um modo geral, a QoS em *frameworks* pode ser resumida em três pontos:

i) *interfaces* pouco flexíveis: as *interfaces* atuais não são geralmente de QoS configurável e fornecem somente um pequeno sub-conjunto de funcionalidades necessárias para o controle e gerenciamento de fluxo de multimídia;

ii) falta de mecanismos para garantias de QoS: é necessário distribuição de controle, gerenciamento e manutenção de mecanismos QoS para que níveis contratados de serviço possam ser confiáveis;

iii) QoS para diferentes condições: é necessário um *framework* de uso geral para aumentar a QoS em diferentes níveis de sistemas e entre diferentes arquiteturas de rede.

Analisando os três pontos citados foram propostas algumas arquiteturas para provisão de QoS com a intenção de definir um conjunto de interfaces configuráveis que forneçam uma estrutura de integração entre controle e mecanismos de gerenciamento de QoS. Um estudo detalhado contendo comparações entre arquiteturas de QoS pode ser encontrado no artigo *A Survey of QoS Architectures* (AURRECOECHEA *et al.*, 1998).

A seguir, serão apresentadas algumas propostas de arquiteturas e de modelos que tratam da QoS em sistemas computacionais.

2.4.1 OMEGA

Na Universidade da Pensilvânia foi proposta uma arquitetura denominada arquitetura OMEGA (NAHRSTEDT & SMITH, 1995). A OMEGA é o resultado de uma pesquisa interdisciplinar que verifica a relação entre requisitos de QoS em aplicações e gerenciamento de recursos locais, como o sistema operacional, e recursos globais, combinando comunicação e recursos remotamente gerenciados para satisfazer essas exigências.

A arquitetura OMEGA assume um subsistema de rede que estabelece limites nos atrasos e possui exigências quanto à largura de banda e um sistema operacional que é capaz de prover garantias de QoS em tempo de execução. A essência da arquitetura OMEGA é a reserva e a gerência de recursos fim-a-fim.

A comunicação é precedida por uma fase onde os requisitos de aplicação são especificados em tempo de execução, isto é, expressos em termos de parâmetros de QoS, negociados, e as garantias são verificadas em vários níveis lógicos, como entre aplicações e subsistemas de rede, aplicações e o sistema operacional, subsistema de rede e o sistema

operacional. São estabelecidas conexões customizadas, resultando na alocação de recursos apropriados para satisfazer os requisitos de QoS da aplicação e a capacidade de rede.

2.4.2 Tenet Architecture

O Grupo Tenet na Universidade da Califórnia propôs em (FERRARI, 1996) uma família de protocolos que examinam experimentalmente uma rede *Asynchronous Transfer Mode* (ATM). A Arquitetura Tenet inclui um Protocolo para Administração de Canal em Tempo Real (RCAP), um Protocolo de Internet em Tempo Real (RTIP) e um Protocolo de Transporte de Mídia Contínua (CMTP).

O RCAP estabelece conexão genérica, reserva de recursos e funções de sinalização para o resto da família do protocolo. O RCAP abrange o transporte e as camadas de rede para toda e qualquer reserva de recursos e de configuração. O CMTP é projetado para oferecer suporte contínuo de mídia. É um protocolo que funciona na parte superior do RTIP, que dá entrega seqüenciada e periódica de amostras contínuas de mídia com controle de QoS, atrasos e limites de erros.

2.4.3 TINA QoS *Framework*

É um *framework* proposto (AURRECOECHEA *et al.*, 1998) para especificar aspectos QoS de telecomunicações distribuídas dentro do contexto de uma arquitetura de computação. O *framework* de QoS trata dos pontos de vista computacionais e de engenharia sobre aplicações de telecomunicações distribuídas. É determinado pela separação entre aplicações de telecomunicação e o Ambiente de Processamento Distribuído (DPE), isto é, serviços multimídia oferecidos por uma fonte utilizam o DPE, a comunicação básica e a capacidade de comunicação.

Do ponto de vista computacional, os parâmetros QoS requeridos para prover garantias a objetos são relacionados como atributos de serviços. Com relação a QoS, as aplicações são

avaliadas por mecanismos de gerenciamento de recursos que são necessários para assegurar as garantias QoS.

2.4.4 End System QoS *Framework*

Na Universidade de Washington, foi proposto um *framework* (GOPALAKRISHNA & PARULKAR, 1994) para oferecer garantias de QoS dentro dos sistemas finais para aplicações multimídia de redes. O *framework* possui quatro componentes principais: especificação de QoS, mapeamento de QoS, execução de QoS e implementação de protocolo.

A especificação QoS está em um alto nível e usa parâmetros para permitir que as aplicações definam os seus requisitos. Baseadas na especificação de QoS, as operações de mapeamento de QoS derivam os requisitos de recursos para cada sessão de aplicação fim-a-fim. Os recursos de sistemas incluem processamento, memória e rede. A terceira parte do *framework* é a execução, que diz respeito principalmente a fornecer garantias de processamento em tempo real para a transferência de mídia. O componente final do *framework* é um modelo de implementação para protocolo do nível de aplicação.

2.4.5 QoSPath

O QoSPath (MATSUI *et al.*, 1999) é um modelo de objetos para especificar, traduzir e arbitrar QoS em sistemas de QoS adaptativos. O modelo QoSPath permite que os usuários e os programas de aplicações especifiquem suas preferências de QoS através do ajuste de QoSPoints. Cada QoSPoint representa um valor que corresponde a um conjunto de parâmetros de QoS. No QoSPath é definido que a função de utilidade recebe um valor de [0,1], onde 0 é a qualidade mínima e 1 é o alvo da QoS.

É possível observar na Figura 2.8 que o QoSPoint{30 fps, 640x480} oferece o valor útil de 1 e que o QoSPoint{5 fps, 160x120} oferece o valor 0. Cada QoSPoint é conectado, da

maior para a menor utilidade ao QoSPath, representando assim um conjunto ordenado de preferências do usuário.

O QoSPoint captura a diversidade dos formatos de QoS do modelo, mas a estrutura do QoSPath permanece a mesma para arbitrar os tipos de QoSPoints. A tradução de um modelo QoSPath é definido pelos seus QoSPoints, sendo convertido o QoSPath do usuário para um QoSPath de sistema.

Uma das principais características do QoSPath é possibilitar que o programa de aplicação defina suas preferências de QoS. Com o QoSPath valores de QoS podem ser especificados por apenas alguns parâmetros, fincando separadas as funções para especificar, traduzir e arbitrar a QoS. O QoSPath também oferece uma interface para notificar os programas de aplicação sobre os valores de QoS configurados.

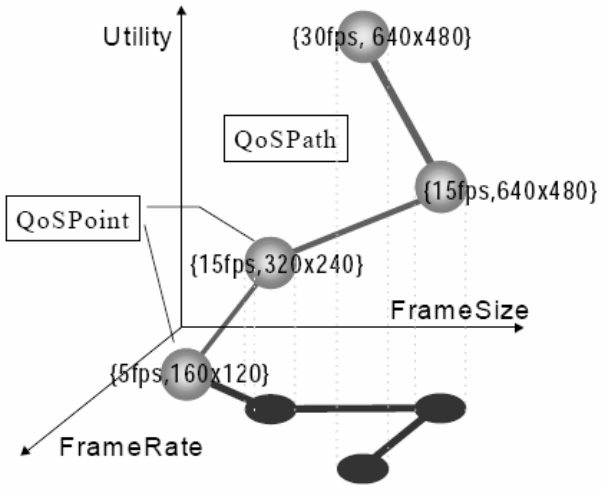

Figura 2.8 QoSPath e QoSPoints [MATSUI *et al.*, 1999]

Para a especificação de QoS podem ser definidos apenas dois QoSPoints, sendo um valor mínimo e um valor alvo. É possível detalhar mais a especificação adicionando QoSPoints ao modelo. O QoSPath especificado pelo usuário é traduzido no nível de sistema.

Os valores úteis de todos os modelos QoSPaths devem ser normalizados para permitirem comparações, sendo que o algoritmo de normalização pode ser definido pelo usuário.

3 PROPOSTA DO TRABALHO

A fim de atender os problemas relacionados com a construção de sistemas supervisórios para dispositivos móveis, este capítulo apresenta a proposta de um *framework*, tema da presente dissertação de mestrado.

3.1 MODELO DE ALTO NÍVEL

A especificação do modelo de alto nível é feita através da definição de uma arquitetura baseada em níveis visando proporcionar um maior grau de flexibilidade e generalidade entre os seus componentes, conforme pode ser observado na Figura 3.1.

Esses níveis estão divididos em:

- Nível 1 – Modelagem: este nível contém uma representação computacional do "mundo real", isto é, inclui os conceitos e os requisitos da aplicação.
- Nível 2 – Visualização: este nível diz respeito às formas de visualização que irão representar o dispositivo de automação a ser supervisionado.
- Nível 3 – Comunicação: este nível se refere ao modo como os dados de supervisão são acessados pelas aplicações, considerando a infra-estrutura de comunicação de dados.

Os níveis são definidos em tempo de projeto. Além disso, os níveis de visualização e comunicação possuem definições dinâmicas, ou seja, que também são consideradas em tempo de execução. Tanto as questões relacionadas com o desenvolvimento dos sistemas supervisórios em tempo de projeto, quanto em tempo de execução, são abordadas por este trabalho.

Após a Figura 3.1 são descritos, brevemente, os componentes presentes na arquitetura proposta, para nas seções seguintes (3.1 a 3.3) apresentar, com detalhes, cada um dos níveis com os seus respectivos componentes:

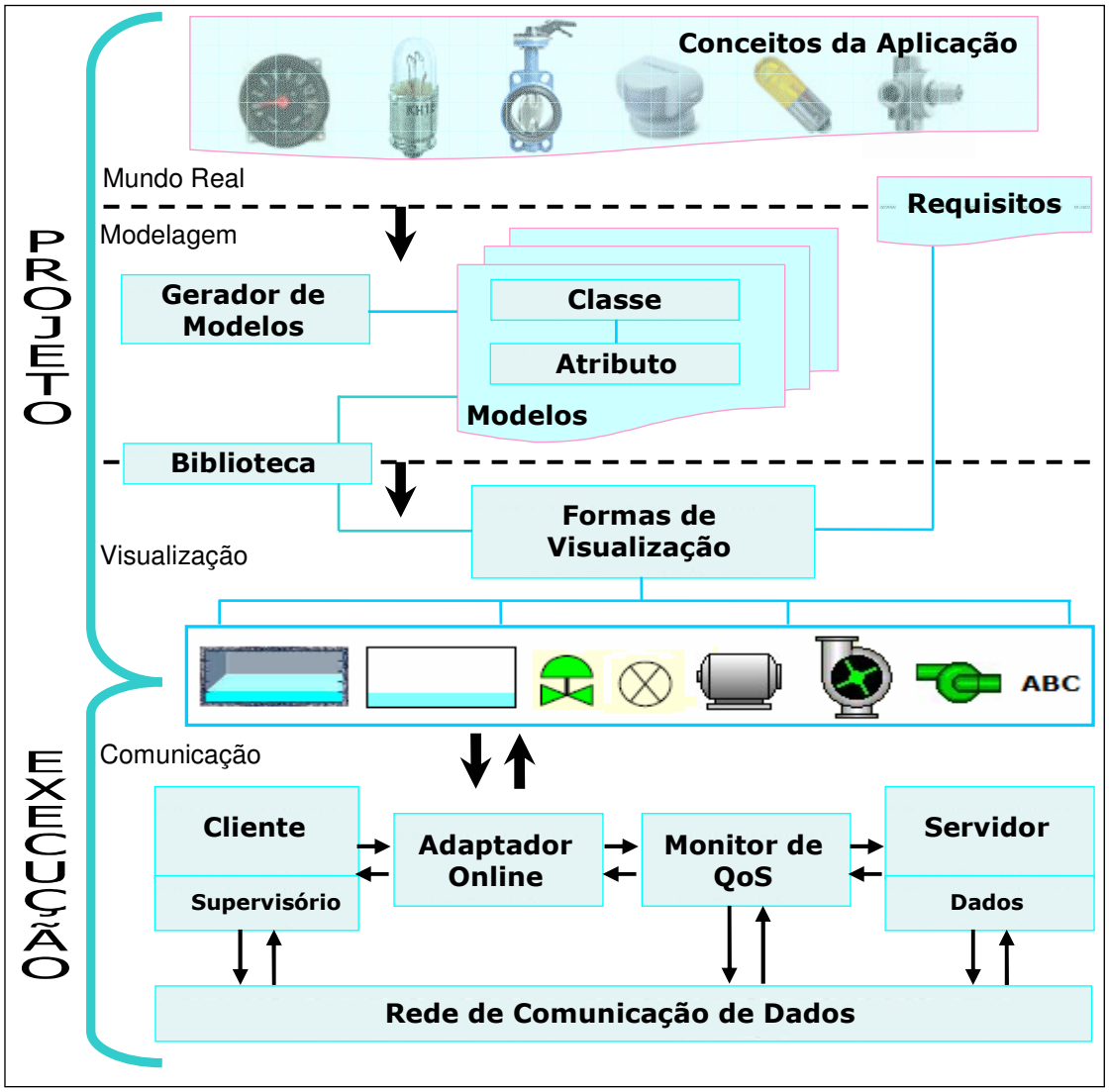

Figura 3.1 Arquitetura Proposta

▪ **Conceitos da Aplicação**: definição do projeto baseado no domínio da aplicação. Considera-se o sistema de automação, os dispositivos e o processo a ser supervisionado, definindo as funcionalidades que o sistema supervisório deverá suportar.

▪ **Biblioteca**: local onde são armazenados os modelos orientados a objetos e as formas de visualização previamente desenvolvidos, visando promover a reusabilidade e acelerar o tempo de desenvolvimento de novas soluções.

▪ **Requisitos**: corresponde à definição dos requisitos de aplicação e de comunicação, tais como: freqüência de atualização dos dados, níveis de visualização e requisitos de QoS.

▪ **Gerador de Modelos**: com a utilização dos conceitos do domínio da aplicação, dos modelos previamente desenvolvidos e da especificação dos requisitos, é criado um novo modelo orientado a objetos para uma aplicação específica.

▪ **Modelo Orientado a Objetos**: é construído para ajudar a compreender aspectos da realidade através de uma representação computacional que utiliza conceitos da orientação a objetos.

▪ **Gerador do Sistema:** ferramenta para desenvolvimento dos sistemas supervisórios que irão ser executados nos dispositivos móveis;

▪ **Aplicação SCADA**: é a aplicação responsável por proporcionar ao usuário a supervisão móvel de um processo, através da comunicação remota e das várias formas de representatividade gráfica.

▪ **Adaptador Online**: responsável por ajustar dinamicamente as formas de visualização na tela do sistema supervisório, de acordo com a QoS oferecida pela infra-estrutura de comunicação de dados.

▪ **Monitor de QoS**: componente responsável por monitorar a qualidade e a disponibilidade dos serviços existentes na rede de comunicação;

▪ **Infra-estrutura de Comunicação**: é a rede de comunicação, na qual são trafegados os dados entre a aplicação geradora e receptora das informações referentes ao processo de supervisão.

▪ **Sistema de Automação** e **Bases de Dados**: são considerados por este trabalho como fontes de dados de supervisão. São nesses componentes que o *framework* proposto deverá buscar as informações necessárias referentes ao processo a ser supervisionado, como os valores das variáveis de processo**.**

3.1.1 Modelagem

A modelagem baseada no paradigma da orientação a objetos é uma técnica para construção de modelos que representam computacionalmente a complexidade do mundo real, especialmente para aplicações industriais. No presente trabalho o sistema de supervisão é modelado através da utilização dos conceitos da orientação a objetos, quais sejam: classes, objetos, atributos e pelos diagramas de classes que é um sub-conjunto de modelos orientados a objetos utilizado para descrever a parte estrutural de um ambiente de supervisão.

Os modelos orientados a objetos correspondentes ao domínio da aplicação podem ser gerados por ferramentas de modelagem, orientada a objetos, que permitam a criação de classes, atributos, métodos e relações entre classes.

Uma visualização do conceito apresentado pode ser observada na Figura 3.2, através de um mapeamento do mundo real para um modelo computacional, cuja classe "Processo" do modelo orientado a objetos contém um conjunto de dispositivos de automação presente no domínio da aplicação. A classe “Processo” contém as classes “Bomba”, “Válvula” e “Reservatório”. Os atributos existentes em cada uma dessas classes representam as suas características. Um dispositivo de automação poderia ter várias especializações como, por exemplo, os dispositivos representados pela classe-filha “Tanque de Água” que são especializações de uma classe mãe chamada “Reservatório”. A partir de uma classe de dispositivos, outras especializações também poderiam ocorrer. O conceito de especialização, presente na orientação a objetos e utilizado neste trabalho, permite identificar classes-mãe, chamadas de gerais, e classes-filha, chamadas de especializadas, e que herdam as características das classes-mãe. A herança é um dos conceitos mais importantes da orientação a objetos, pois permite que a classe-filha possa reaproveitar os atributos e métodos declarados na classe-mãe, reduzindo o tempo de desenvolvimento de sistemas computacionais e, nesse caso, os sistemas supervisórios.

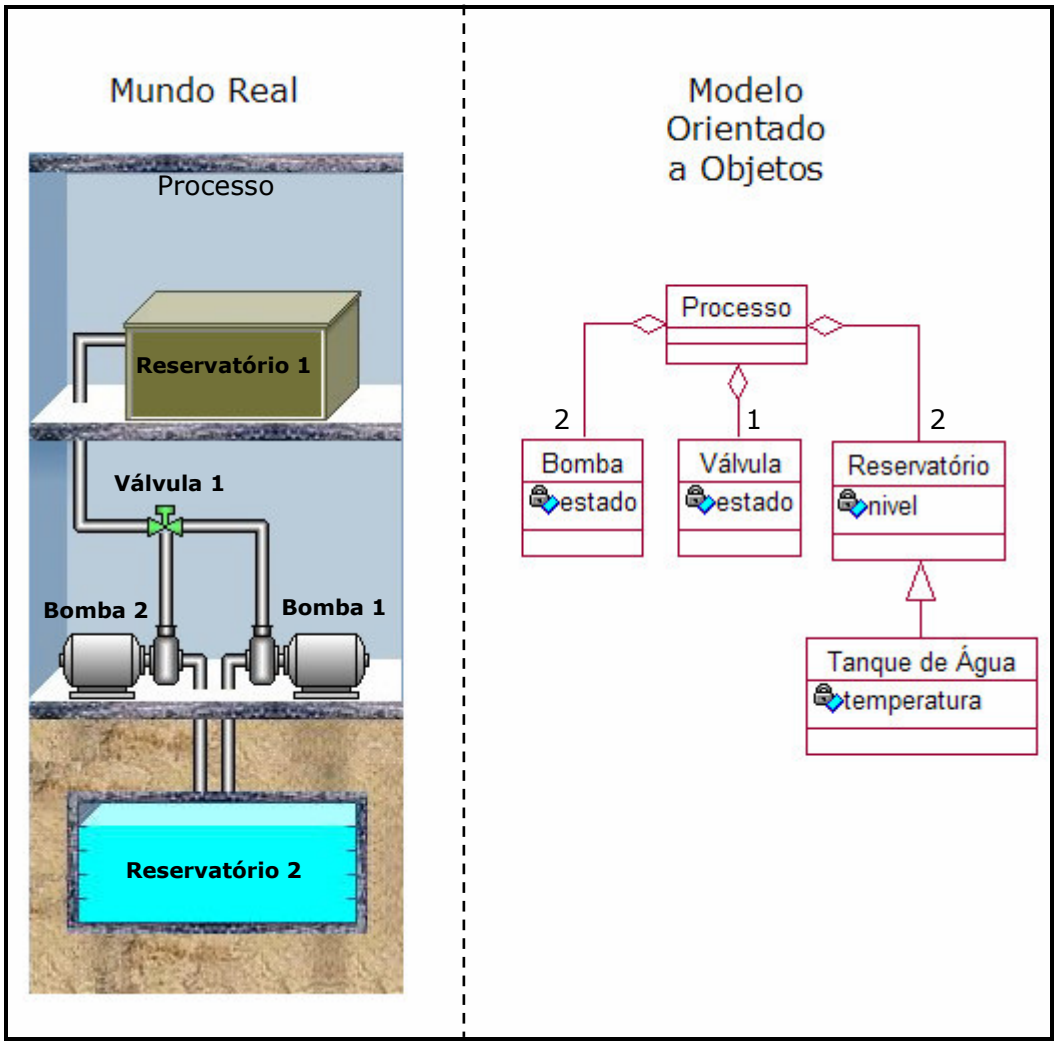

Figura 3.2 Mapeamento do Mundo Real para um Modelo Computacional

Além da associação entre as classes do modelo orientado a objetos com os dispositivos de automação presentes no mundo real, devem, também, estar associados ao mapeamento, os requisitos referentes ao domínio da aplicação. Os requisitos formam um conjunto de classes para especificações e métricas correspondentes às definições de valores como: freqüência com que os dados devem ser coletados no sistema de automação e atualizados no sistema de supervisão, tempo máximo para acesso de informações e definição dos requisitos de QoS, estando relacionadas as formas de visualização com a qualidade e a disponibilidade dos serviços oferecidos pela infra-estrutura de comunicação de dados. A Figura 3.3 ilustra o conjunto de classes referentes aos requisitos.

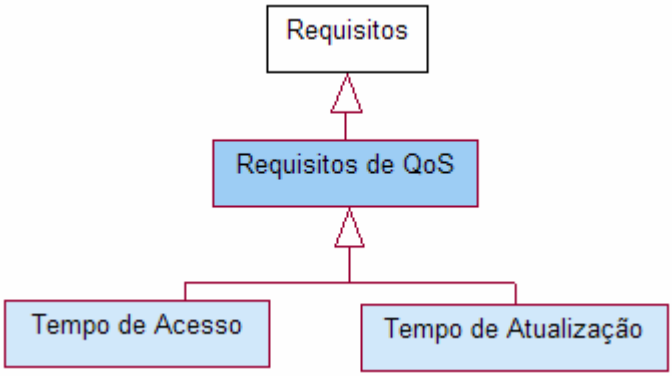

Figura 3.3 Especificação de Requisitos

Diferentes formas de visualização nas telas de supervisão podem ser relacionadas às classes correspondentes aos dispositivos de automação. Essas classes podem estar associadas com requisitos de QoS para acesso e atualização dos dados de supervisão. Uma possível solução para integrar os requisitos de QoS com as formas de visualização seria permitir que a aplicação de supervisão definisse os seus requisitos de QoS. No levantamento dos requisitos envolvidos no domínio da aplicação devem ser consideradas as características, as funcionalidades e as prioridades dos objetos que estarão presentes em uma tela de supervisão, pois esses objetos podem, não necessariamente, exigir o mesmo tempo de atualização do seu estado ou, ainda, um objeto poderia ter prioridade de visualização superior a outro. A prioridade de visualização é um valor (alto, médio ou baixo) que está associado a cada dispositivo de automação a ser supervisionado, permitindo definir qual visualização é mais importante que outra. Para exemplificar a idéia pode-se considerar o caso de que numa planta industrial qualquer, um tanque de água poderia oferecer dois atributos para supervisão: nível e temperatura. Nesse processo podem existir vários requisitos para supervisão de tal processo, porém, nesse exemplo, é considerada apenas a seguinte hipótese:

- O nível e a temperatura do tanque de água podem trocar de estado ao mesmo tempo ou em tempos diferentes. Porém, a QoS presente na infra-estrutura de comunicação de dados pode não conseguir atender aos requisitos de QoS definidos pela aplicação de supervisão, ou conseguir atender parcialmente, levando a exibição apenas de um dos atributos. Nesse caso

seria exibido o atributo que foi definido como sendo de maior prioridade de visualização, exigindo do sistema supervisório uma adaptação dinâmica.

A hipótese exemplifica uma situação que deve ser considerada em tempo de projeto, no levantamento dos requisitos da aplicação.

3.1.2 Visualização

Após o mapeamento computacional, considera-se a existência de uma associação entre as classes e atributos do modelo orientado a objetos com as possíveis formas de visualização que irão representar graficamente o dispositivo de automação a ser supervisionado.

As formas de visualização são representações gráficas que se referem aos dispositivos de automação. Essas formas de visualização podem ser utilizadas para mostrar, através de imagens, como determinados dispositivos de automação e variáveis de processos (representados computacionalmente por classes e atributos) trocam de estado ao longo do tempo.

Numa tela de supervisão podem existir vários indicadores visuais com diferentes cores e formas para se obter uma fácil comparação entre variáveis supervisionadas, sendo obtidas em tempo de execução do processo ou até mesmo através de um histórico arquivado que pode ser utilizado para análise de tendências de processos e monitoramento de eficiência da produção. Através da representação gráfica o processo de supervisão pode se tornar muito mais prático e rápido ao operador, permitindo a visualização de informações a respeito do estado de um dispositivo de automação ou processo produtivo. Por exemplo: ao invés de um simples piscar de lâmpadas o operador pode ter uma melhor visualização quando efetivamente enxerga o abrir de uma válvula ou o acionamento de um motor.

Conforme pode ser observado na Figura 3.4, as formas de visualização são compostas por elementos de visualização que podem assumir diferentes modos para representação de

uma mesma informação. As formas de visualização são definidas pela aplicação, em tempo de projeto, mas também pelo Adaptador Online, em tempo de execução do sistema supervisório, que seleciona a visualização em função dos requisitos de QoS. Quanto às formas de visualização, elas podem ser do tipo:

▪ Bitmap: que permite a inserção de imagens na tela de supervisão;

▪ Animação: que possibilita a inserção de imagens na tela do dispositivo de supervisão, com características de formar um conjunto de imagens estáticas, apresentadas em seqüência, para representar um movimento;

▪ Bargraph: que permite a visualização dos dados supervisionados em formato de barras verticais ou horizontais;

▪ Texto: permite a representação de dados através de um conjunto de caracteres. É uma forma de visualização mais simplificada.

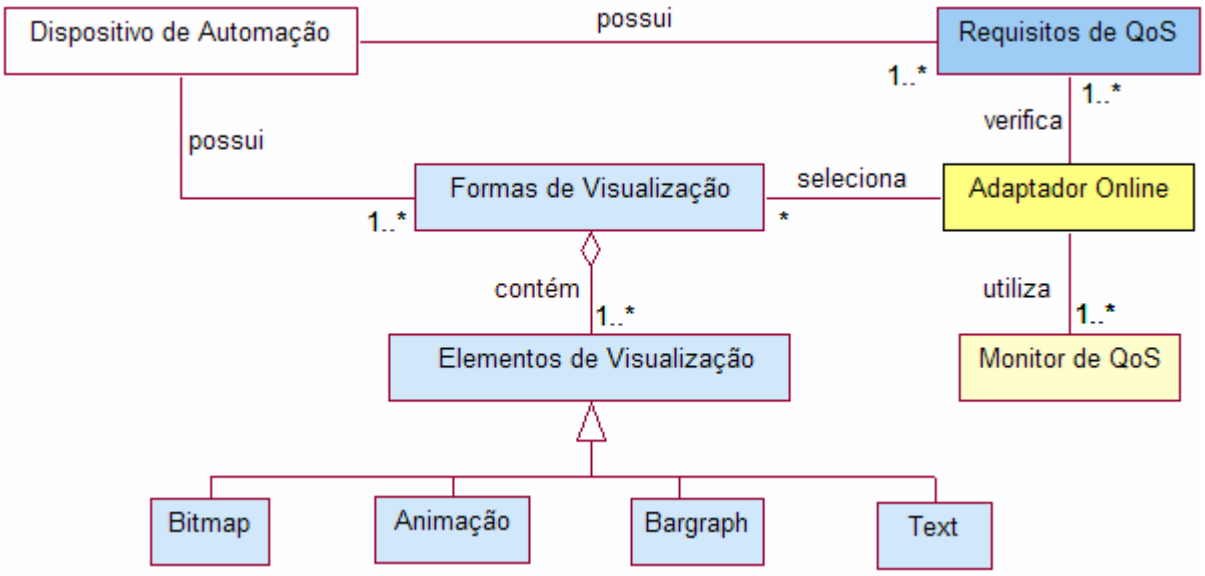

Figura 3.4 Formas de Visualização

Outros tipos de representações gráficas podem ser criados a partir da especialização da classe "Elementos de Visualização" como, por exemplo, uma classe "Gauge" para visualização de dados através de ponteiros. Dessa forma, o *framework* já teria capturado as principais decisões de projeto comuns ao domínio da aplicação, o que reduziria o tempo gasto

com o desenvolvimento de novas soluções, podendo ser adicionada ao *framework* uma nova classe de representação gráfica, através da especialização da classe "Elementos de Visualização" já existentes, conforme ilustra a Figura 3.5.

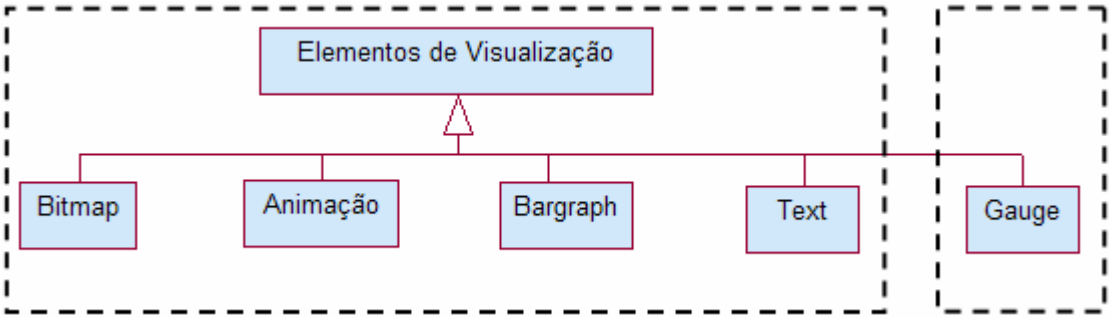

Figura 3.5 Customização de Visualização no *Framework*

Com a especialização de classes, o *framework* pode ser customizado para uma determinada aplicação do domínio, através da criação de subclasses específicas para aquela aplicação. A inserção de classes permite que o *framework* evolua de acordo com as necessidades das aplicações a serem desenvolvidas, enfatizando a reutilização de projetos. Como resultado dessa evolução seria possível construir aplicações mais rapidamente e com estruturas similares.

As formas de visualização podem ser definidas por dois modos:

i) Em tempo de projeto: são definidas as diferentes formas de visualização que podem ser utilizadas para apresentação de um dispositivo de automação. Por exemplo: o nível de um tanque pode ser representado, graficamente, através de um Bargraph ou através de uma animação por imagens ou, ainda, representado em modo texto, através de um conjunto de caracteres, conforme pode ser observado na Figura 3.6. A escolha dessas diferentes formas de visualização dependerá dos requisitos de QoS definidos pela aplicação e pela QoS oferecida pela infra-estrutura de comunicação e de execução.

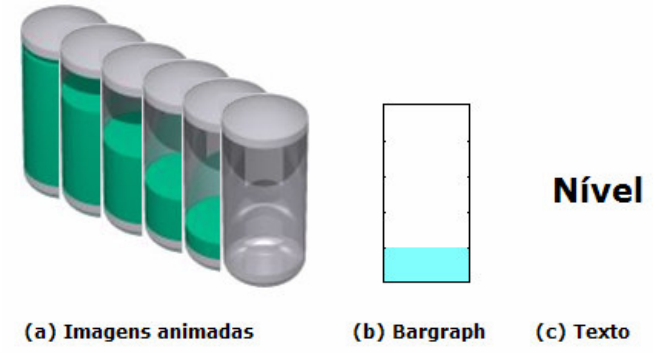

Figura 3.6 Representações de uma mesma informação

ii) Em tempo de execução: no qual são verificados os níveis de QoS oferecidos pela infra-estrutura de comunicação de dados e comparados com os requisitos de QoS atribuídos, em tempo de projeto, para um dispositivo de automação e a sua respectiva forma de visualização. Com os resultados dessa comparação o sistema supervisório realiza uma adaptação dinâmica para visualização das informações;

A especificação dos requisitos de QoS é baseada nas categorias de QoS sugeridas na arquitetura para serviços *web* proposta em (TIAN *et al.*, 2003), na qual os clientes podem definir seus requisitos de QoS, através de parâmetros tais como: tempo de processamento, tempo de resposta e disponibilidade de serviços, sendo a QoS diferenciada pelas categorias platina, ouro e bronze. No presente trabalho, os requisitos de QoS também são definidos por três níveis de representatividade, apresentados na Tabela 1, que variam de acordo com a qualidade e disponibilidade dos serviços oferecidos pela infra-estrutura de comunicação de dados, podendo também variar de acordo com o ambiente de execução do sistema supervisório. A QoS é monitorada sendo atribuídos os seguintes níveis ou classes de serviço:

Tabela 1 Níveis de QoS

Nível	**Descrição**
Ouro	indica que a QoS está com um nível alto
Prata	indica que a QoS está em um nível intermediário
Bronze	indica que a QoS está com um nível baixo

A especificação dos requisitos de QoS são mapeados para a aplicação através da relação entre valores temporais e níveis de QoS, sendo que para cada nível de QoS é configurado um valor de tempo correspondente. Nesse caso, uma aplicação estaria com um nível de QoS Ouro, quando o tempo máximo obtido na comunicação de dados da rede fosse inferior a t1; nível de QoS Prata, quando o tempo máximo estivesse entre t1 e t2; e, nível de QoS Bronze, quando o tempo máximo obtido na comunicação fosse superior a um tempo t3. Baseado nesses níveis são definidas as formas de visualização que um dispositivo de automação irá assumir em cada estado de QoS encontrado na rede de comunicação de dados.

A principal contribuição para a divisão de QoS em níveis é que, mesmo em situações de degradação de QoS da rede, o sistema supervisório seja capaz de dinamicamente adaptar a visualização associada a um dispositivo de automação, permitindo que os usuários tenham uma visão inteligível do processo de supervisão. Sendo assim, se o nível de QoS for alto, nível “Ouro”, é disponibilizado ao usuário uma tela de supervisão completa, com informações textuais e todos os objetos gráficos disponíveis referentes ao processo. Quando houver uma degradação do nível de QoS oferecido, a tela de supervisão disponibilizada para o usuário é parcial, sendo que o sistema considera apenas os objetos de visualização mais relevantes. Porém, se os parâmetros de QoS recebidos estiverem com nível baixo, nível “Bronze,” o usuário terá disponível uma tela de supervisão simplificada, na qual o sistema supervisório apresentará apenas as informações mais significativas do processo ou dos dispositivos de automação, sendo descartada a parte de representação gráfica.

Para exemplificar o conceito, o mesmo nível de um tanque pode ser visualizado de três modos distintos: (i) através da animação por imagens, quando a QoS estiver com o nível alto; (ii) através de um Bargraph, quando a QoS estiver com nível intermediário; e (iii) através de informações textuais, quando a QoS estiver com nível baixo.

Essa classificação por categorias de QoS permite que a visualização das informações seja dada em função da QoS oferecida pela rede de comunicação de dados e pelo ambiente de execução, conforme pode ser observado na Figura 3.7. O objetivo é explorar o máximo dos recursos disponíveis a fim de proporcionar ao usuário alguma informação significativa do processo dentro de um requisito temporal pré-estabelecido, mesmo quando a QoS estiver com um nível baixo.

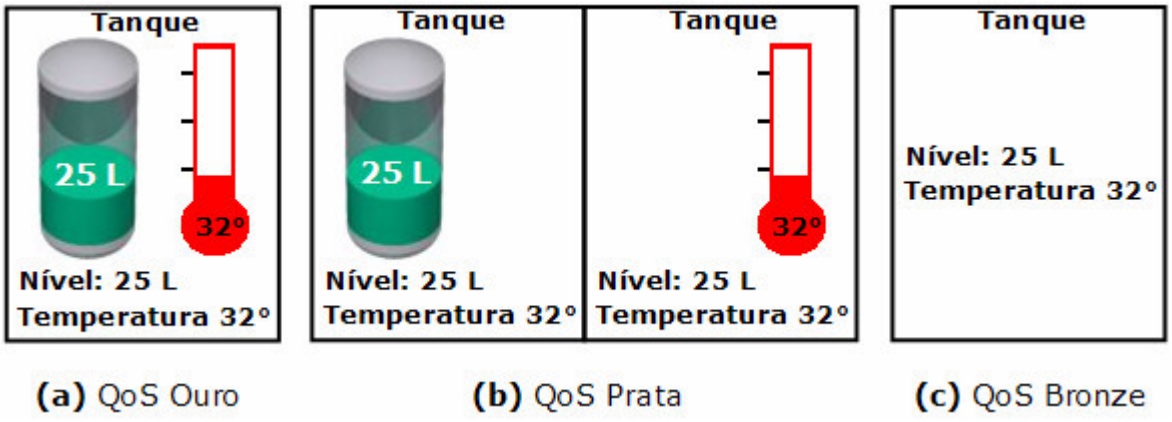

Figura 3.7 Visualização em função da QoS

No exemplo da Figura 3.7(b) a QoS está com um nível "Prata", sendo que existem dois atributos do dispositivo tanque para serem exibidos: nível e temperatura. No entanto, como o nível de QoS "Prata" indica que houve uma degradação de QoS, apenas um dos atributos conseguiria ser representado graficamente dentro do tempo estabelecido. Nesse caso a escolha de qual atributo será exibido com recursos gráficos é dada pela condição de prioridade, ou seja, o atributo que foi definido em tempo de projeto como sendo de maior prioridade de visualização em relação a outro.

3.1.3 Infra-estrutura de Comunicação de Dados

Quatro aspectos serão considerados na análise da infra-estrutura de comunicação de dados proposta no presente trabalho: (i) definição do modelo de comunicação; (ii) formatos dos dados; (iii) acesso e disponibilização das informações; e (iv) monitoramento de QoS.

3.1.3.1 Modelo de Comunicação

A comunicação de dados entre a planta industrial e o dispositivo móvel que executa o sistema supervisório, envolve algumas funcionalidades e fluxos de informações existentes nessa comunicação. A Figura 3.8 identifica alguns módulos que oferecem suporte a definição de um modelo de comunicação.

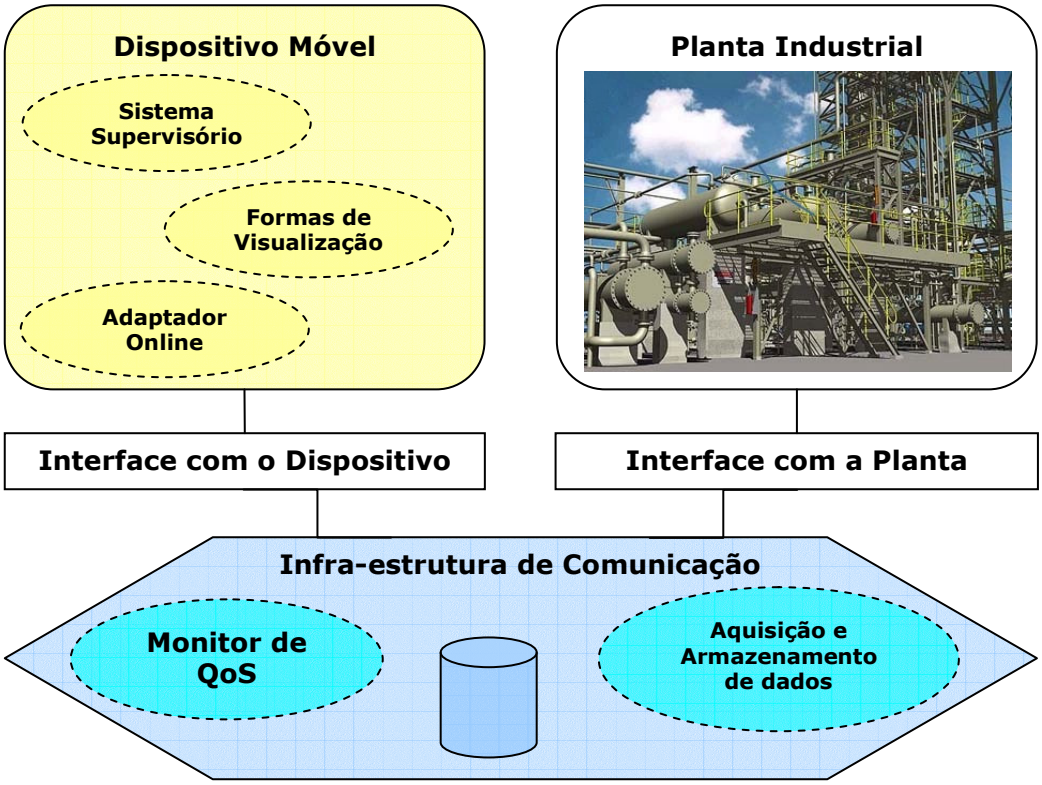

Figura 3.8 Identificação de módulos

A partir da Figura 3.8 são observados os módulos envolvidos na comunicação de dados, sendo responsáveis pela aquisição e armazenamento de dados, pelo monitoramento de QoS, pelas i*nterfaces* de comunicação da planta industrial/dispositivo móvel e pela adaptação de visualização dos dados em função da QoS.

A Comunicação entre os módulos poderia ocorrer por diversas maneiras, dentre elas, uma possível interação seria iniciada pelo dispositivo móvel, no qual o sistema supervisório solicitaria os dados de supervisão para um módulo de aquisição, através de uma interface de comunicação. O módulo de aquisição, por sua vez, realizaria a

comunicação com a planta industrial devolvendo a solicitação ao sistema supervisório. Já a visualização dos dados solicitados pelo sistema de supervisão seria adaptada em função da QoS oferecida pela infra-estrutura de comunicação.

Existem diferentes formas de estabelecer comunicação entre dispositivos fonte e destino. Qualquer um dos dispositivos envolvidos pode assumir um papel diferente nessa comunicação, por exemplo: num modelo cliente-servidor os dispositivos móveis poderiam tanto ser cliente, realizando a leitura dos dados de supervisão em um computador remoto, quanto servidor, disponibilizando os dados de supervisão lidos para outras aplicações. Outro modelo de comunicação poderia ser o *publish/subscribe* que é baseado na troca assíncrona de mensagens e que consiste num conjunto de dispositivos que publicam eventos, os quais são encaminhados para clientes que registraram interesse em recebê-los.

Dentre os possíveis modelos de comunicação optou-se neste trabalho pelo modelo cliente-servidor, no qual o servidor aguarda por conexões e solicitações dos clientes, executando serviços e retornando os resultados das solicitações conforme ilustra a Figura 3.9. Basicamente, o servidor é responsável pela manutenção da informação e o cliente responsável pela leitura dos dados. O cliente lê os dados do servidor, porém é o servidor que coleta os dados na planta industrial, em bases de dados ou diretamente no sistema de automação, tratando e disponibilizando esses dados aos clientes.

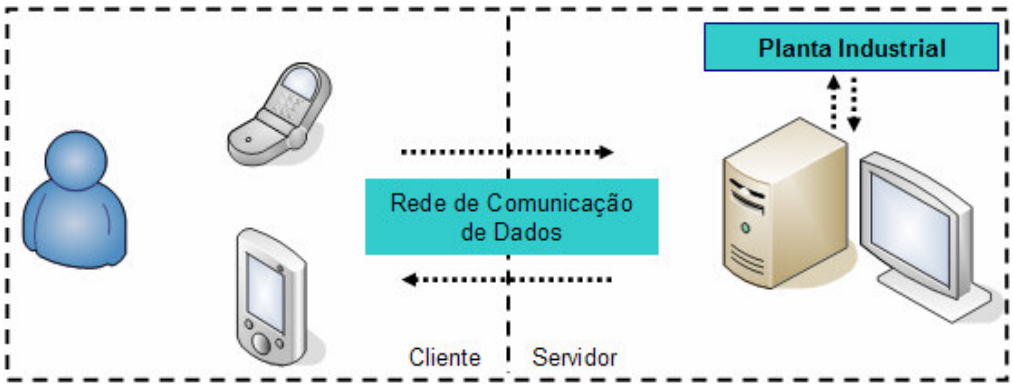

Figura 3.9 Modelo de Comunicação

Em relação ao *hardware* envolvido no modelo de comunicação é atribuída ao servidor uma plataforma computacional baseada em computador pessoal e, ao cliente, uma plataforma baseada em computação móvel. Já a comunicação dos dados, entre as estações cliente e servidor, pode ser realizada tanto através da Internet, quanto através de uma rede local sem fio (*Wireless Local Area Network* - WLAN).

Assim, um dispositivo móvel, com suporte para comunicação de dados em redes, solicita informações de supervisão ao servidor, através da utilização de uma rede de comunicação de dados disponível. A comunicação de dados entre cliente e servidor pode ser combinada de vários modos: um dispositivo móvel PDA acessando os dados do servidor tanto via Internet quanto através de uma rede local, conforme o diagrama de classes apresentado na Figura 3.10.

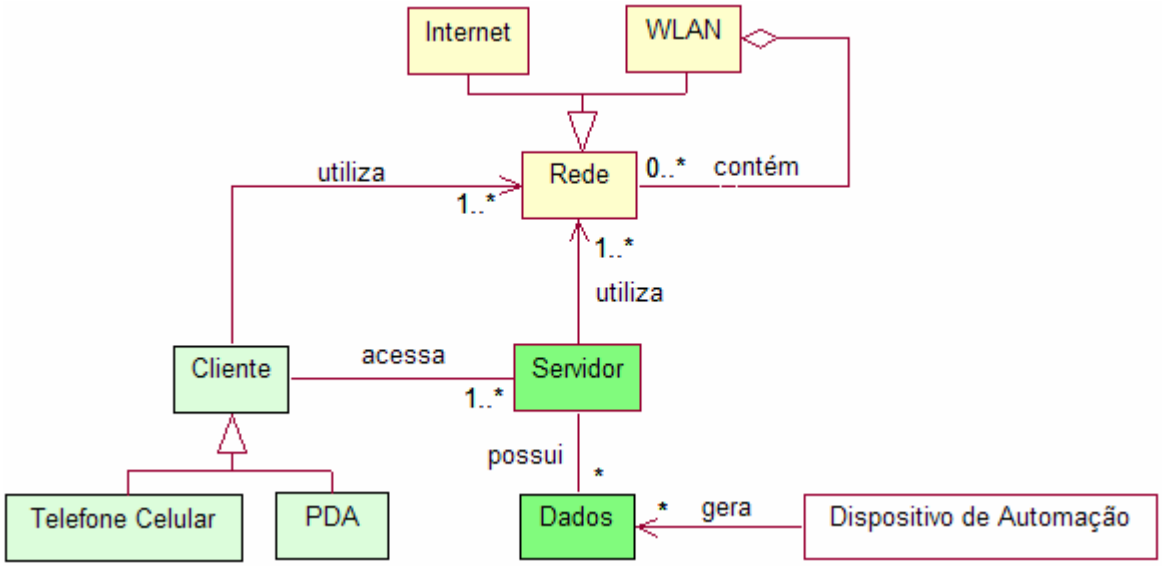

Figura 3.10 Classes do Modelo de Comunicação

3.1.3.2 Formato dos Dados

É no servidor, o local onde ficam disponíveis os dados de supervisão correspondentes aos valores dos atributos das classes que representam os dispositivos de automação. Esses dispositivos poderiam ser representados no sistema supervisório através de diferentes mídias, tais como: áudio, vídeo ou gráficos. No entanto, são considerados neste trabalho os seguintes formatos de mídia:

i) imagens: são as visualizações gráficas referentes às variáveis dos dispositivos de automação e do processo de supervisão. As visualizações gráficas ficam disponíveis no servidor quando os recursos de *hardware* do dispositivo móvel são limitados como, por exemplo, memória insuficiente para o armazenamento dos dados;

ii) texto: são informações formadas por um conjunto de caracteres que contém os valores das variáveis referentes aos dispositivos de automação e ao processo de supervisão.

Tanto a imagem quanto o texto possuem, no seu contexto, informações com o mesmo significado. Porém, o que diferencia um tipo de mídia de outra é a sua representatividade. O nível de água num tanque poderia ser, por exemplo, representado graficamente ou através de um conjunto de caracteres.

3.1.3.3 Acesso e Disponibilização dos Dados

Os dados de supervisão acessados pelas estações cliente são adquiridos a partir de uma aplicação, instalada no servidor, que acessa o sistema de automação ou uma base de dados com os valores das variáveis de supervisão, conforme pode ser observado na Figura 3.11. Assim, essa aplicação que é executada no servidor, pode ser definida como um coletor de dados, o qual permite integrar os sistemas supervisórios dos dispositivos móveis criados pelo *framework* proposto, com sistemas de automação ou com outros sistemas supervisórios disponíveis no mercado.

Essa integração pode ser feita através da leitura das variáveis de supervisão diretamente em um sistema de automação ou em uma base de dados referente a essas variáveis. Quanto à leitura de variáveis no sistema de automação, o coletor de dados poderia implementar a comunicação e trocar informações diretamente no sistema de automação, o que poderia ocorrer via protocolos HTTP ou OPC. Com relação à leitura das variáveis em uma base de dados, o coletor de dados poderia implementar a comunicação com o banco de dados específico para uma aplicação.

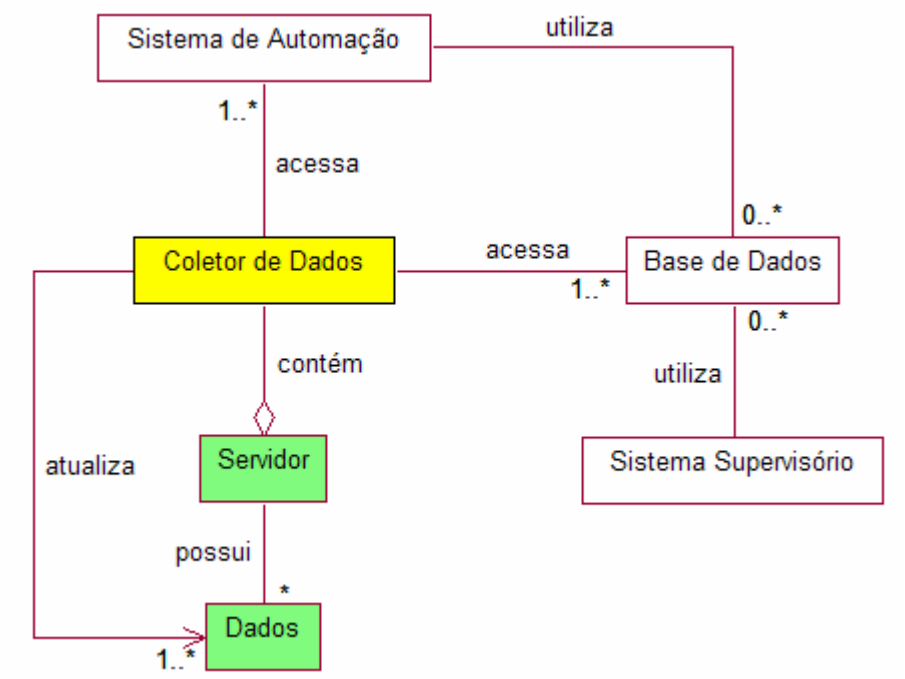

Figura 3.11 Coleta e Disponibilização de Dados

O dispositivo cliente pode acessar diretamente a base de dados ou o sistema de automação, mas isso aumentaria a carga de processamento no dispositivo móvel e também poderia aumentar a quantidade de dados que trafegam pela rede de comunicação. Um exemplo prático é considerar que a supervisão de um determinado processo é efetuada por um telefone celular, utilizando a Internet para se comunicar com o servidor, tendo a seguinte situação: como o acesso à Internet efetuado pelo telefone celular é tarifado pelas operadoras de telecomunicações com base na quantidade de dados transmitidos, quanto mais dados trafegarem pela rede de comunicação, maior é o custo para a supervisão do processo.

Desse modo, há preferência por coletar, tratar e disponibilizar as informações de supervisão diretamente no servidor, através do coletor de dados, que resulta, por parte do cliente, na leitura de um dado de supervisão já pronto para exibição, sem a necessidade de tratá-lo novamente, pois esse serviço já foi executado no servidor. Isso leva a uma redução do custo de processamento no cliente e menor tráfego de dados na rede de comunicação.

3.1.3.4 Monitor de QoS

A QoS envolve questões desde a segurança até a monitoração de serviços. Isso inclui mecanismos para definir como os dados com diferentes requisitos compartilham o mesmo meio de comunicação, ou seja, a distinção de dados mais e menos prioritários.

A monitoração de QoS deveria também incluir o ambiente de execução do sistema supervisório, medindo assim o atraso fim-a-fim desde a chamada do procedimento de envio de uma mensagem até a efetiva recepção desses dados pelo sistema supervisório executado no dispositivo móvel.

Poderia ser monitorado o tempo de processamento da plataforma alvo para realizar uma determinada tarefa, somando esse tempo aos tempos de comunicação da rede para computar uma QoS total (processamento + comunicação).

Existem dois métodos básicos para medir a QoS numa rede de comunicação: (i) método passivo, que mede o comportamento da rede através da observação da taxa de pacotes num sistema final, sendo que as métricas são obtidas pelo comportamento da aplicação que é imposto pela rede de comunicação; e (ii) método ativo, que insere e coleta pacotes na rede de comunicação permitindo que a estação de medição possa obter parâmetros tais como: alcançabilidade, tempo de resposta na transmissão dos dados e perda de pacotes.

Existem diversas alternativas para determinação das métricas de QoS, tais como o retardo médio, o *jitter* e as variações de comunicação.

Dentre as alternativas para medição, optou-se pelo uso da técnica *Round-Trip Time* (RTT), que mede o tempo de ida e volta de um pacote, ou seja, mede o tempo gasto para uma mensagem ir do cliente ao servidor e retornar do servidor ao cliente, conforme ilustra a Figura 3.12.

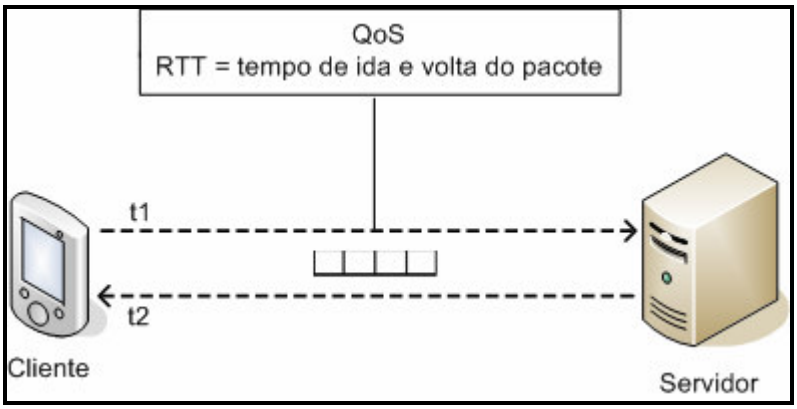

Figura 3.12 Monitoramento de QoS com RTT

Com os valores de RTT obtidos pelo monitor de QoS em tempo de execução podem ser definidos os intervalos de QoS existentes na rede de comunicação, pois seria possível definir uma faixa de operação para a variação da QoS, na qual o menor RTT medido corresponderia ao nível de QoS "Ouro" e o maior RTT medido corresponderia ao nível de QoS "Bronze", sendo que essa faixa de operação seria divida pelo nível "Prata". Ou seja, cada valor de RTT mensurado seria atribuído a um dos três níveis existentes na faixa de operação.

O valor de RTT obtido através do tempo total para que um pacote deixe o cliente, atinja o servidor e retorne, poderia ser calculado para uma ou para muitas mensagens com eventuais variações no tamanho delas, sendo obtido um RTT médio.

Após se ter definido o nível de QoS atual (Ouro, Prata e Bronze), através da medição e comparação do RTT, um adaptador de visualização ajusta, dinamicamente, qual a qualidade e a forma de visualização que um dispositivo de automação irá assumir na tela de supervisão.

Também existem diversas alternativas para definir o instante de medição do RTT: (i) antes de ocorrer uma comunicação para acesso aos dados de supervisão; (ii) no instante da comunicação; ou (iii) periodicamente. A QoS da rede de comunicação também pode ser obtida através de um processo independente de monitoração, que roda no cliente, no servidor ou em ambos. Desse modo, esse processo informaria, periodicamente, a QoS atual aos solicitantes. Outra possibilidade seria medir a QoS da rede de comunicação sempre que uma variável de supervisão precisasse ser atualizada. Logo, antes do cliente solicitar ao servidor as

formas de visualização de um processo de supervisão, o monitor verifica qual a QoS da rede. Cada dispositivo de automação a ser supervisionado terá requisitos temporais associados,como um sensor de temperatura, que pode variar o seu estado aleatoriamente, mas o tempo de comunicação e visualização do estado do sensor pode ser determinado com base no domínio da aplicação.

3.2 IMPLEMENTAÇÃO

Para implementação da arquitetura computacional anteriormente mencionada, optou-se pelo *Java 2 Micro Edition* (J2ME), que é uma edição da linguagem de programação Java direcionada a dispositivos com recursos variados consistentes em memória, processamento e comunicação (MUCHOW, 2004). A escolha de gerar as aplicações de supervisão em linguagem Java se deve ao fato dela oferecer portabilidade do código gerado, através de *bytecodes* que são interpretados pela máquina virtual Java, ou *Java Virtual Machine* (JVM). Isso permite que uma mesma aplicação de supervisão gerada possa ser utilizada por uma variedade de outras plataformas computacionais, conforme ilustra a Figura 3.13.

Figura 3.13 Variedade de Dispositivos Móveis

Outra escolha para a implementação é o uso da especificação XMI - XML *Metadata Interchange*, criada pela *Object Management Group* (OMG, 2006) com a intenção de permitir

a troca de informações entre ferramentas de modelagem que implementam a UML e amplamente difundida. A XMI é baseada na linguagem XML e seu objetivo é permitir a troca de modelos UML entre as ferramentas de modelagem (PENDER, 2004). Além disso, com o arquivo de saída XMI gerado por uma ferramenta UML, outros *softwares* podem utilizá-lo para criar código em uma determinada linguagem de programação, conforme a Figura 3.14.

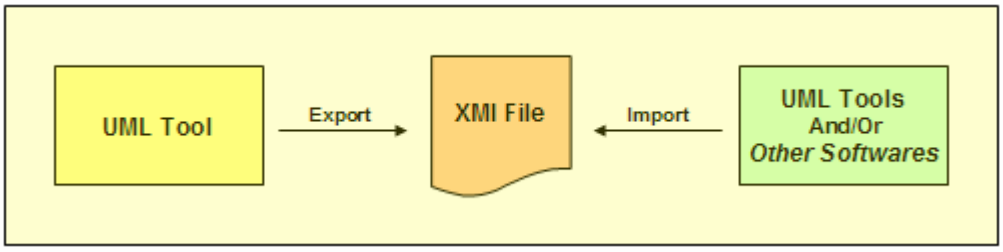

Figura 3.14 Troca de Informações com XMI

3.2.1 Ferramentas Desenvolvidas

A partir da arquitetura baseada em níveis, apresentada na seção 3.1, é definida uma arquitetura de implementação, ilustrada na Figura 3.15, onde estão presentes os conceitos utilizados por este trabalho e as tecnologias envolvidas na implementação da arquitetura. Para a materialização da proposta do *framework*, são construídas duas ferramentas: uma para a geração dos sistemas supervisórios e outra responsável pela comunicação dos dados de supervisão entre cliente (dispositivo móvel) e servidor.

A ferramenta para geração dos sistemas supervisórios, denominada de Ferramenta de Geração, implementa as seguintes funcionalidades:

▪ **Funcionalidade 1**: leitura de modelos orientados a objetos da aplicação, através da importação de arquivos XMI;

▪ **Funcionalidade 2**: leitura e geração de arquivos XML para possíveis novas classes de dispositivos. Essa opção permite ao projetista criar uma nova classe diretamente na Ferramenta de Geração do sistema supervisório, descrevendo-a em XML, sem a necessidade da ferramenta de modelagem;

▪ **Funcionalidade 3**: armazenamento de projetos na biblioteca de projetos;

▪ **Funcionalidade 4**: reuso dos projetos armazenados na biblioteca de projetos, acelerando o tempo de desenvolvimento de uma nova solução;

▪ **Funcionalidade 5**: seleção das formas de visualização desejadas para as classes e os atributos do projeto em questão;

▪ **Funcionalidade 6**: suporte para a especificação de requisitos;

▪ **Funcionalidade 7**: permite a geração automática de código Java para dispositivos móveis com suporte a J2ME. Essa opção transforma o projeto numa aplicação de supervisão (corresponde ao módulo "cliente" da arquitetura proposta);

▪ **Funcionalidade 8**: geração de arquivo para o servidor, contendo a especificação dos dados de supervisão. Essa opção permite que o servidor conheça quais as variáveis de processo serão supervisionadas, efetuando a coleta e a disponibilização dos dados.

De outra parte, a ferramenta responsável pela comunicação dos dados entre cliente e servidor é chamada de **Coletor de Dados** e contém as seguintes funcionalidades:

▪ **Funcionalidade 1**: leitura do arquivo de especificação gerado pela Ferramenta de Geração do sistema supervisório;

▪ **Funcionalidade 2**: acesso de leitura ao banco de dados que contém os valores das variáveis de supervisão;

▪ **Funcionalidade 3**: atualização e disponibilização das variáveis de supervisão, através da geração de um arquivo de dados no servidor HTTP;

▪ **Funcionalidade 4**: comunicação com outros sistemas supervisórios, através de bases de dados;

▪ **Funcionalidade 5**: possibilidade de comunicação diretamente com determinados sistemas de automação. Nessa opção, poderia ser implementada uma comunicação através de OPC.

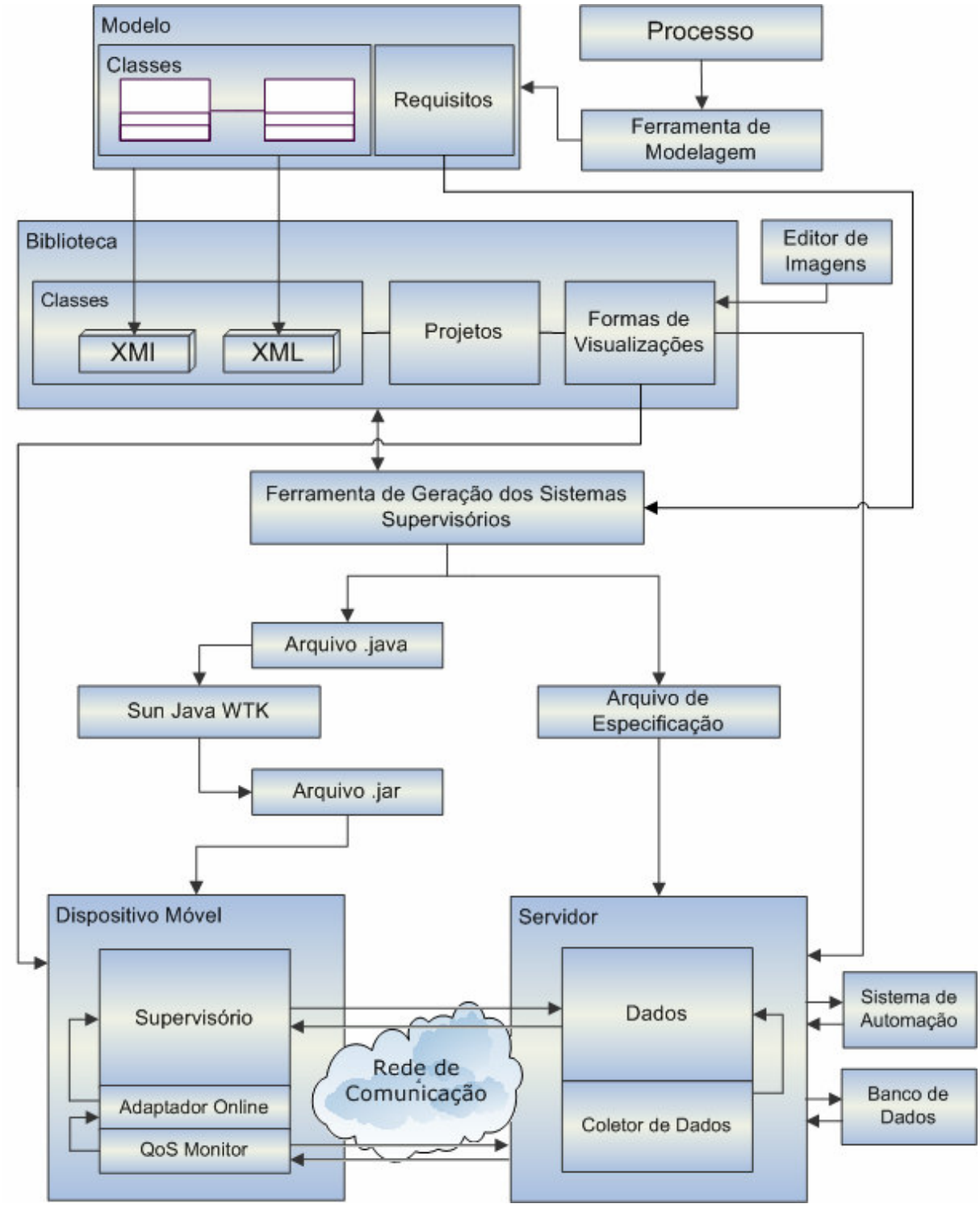

Figura 3.15 Arquitetura de Implementação

3.2.1.1 Ferramenta de Geração

A ferramenta construída para gerar os sistemas supervisórios é desenvolvida utilizando o ambiente de programação Delphi 7. Essa ferramenta construída permite desde a importação de arquivos até a geração automática de código em linguagem Java. Com relação à importação de arquivos, tal ferramenta oferece suporte a dois tipos de arquivos:

(i) Arquivos XMI: correspondentes aos modelos orientados a objetos da aplicação de supervisão a ser projetada, contendo classes, atributos e relacionamentos;

(ii) Arquivos XML: correspondentes a classes e atributos criados pelo projetista na própria Ferramenta de Geração.

A Figura 3.16 apresenta a janela inicial da Ferramenta de Geração e na Figura 3.17 é exibida a janela que corresponde ao ambiente de desenvolvimento das aplicações de supervisão, com uma visualização de menus, campos para edição de dados, opções de configuração, seleção de um catálogo de visualizações, importação e criação de classes, importação de modelos e demais funcionalidades que serão descritas a seguir.

Os objetos visuais presentes no ambiente de desenvolvimento das aplicações permitem realizar os seguintes procedimentos:

- abrir projetos existentes;
- salvar projetos na biblioteca;
- importar classes no formato XML;
- criar uma nova classe;
- importar um arquivo XMI do modelo orientado a objetos da aplicação;
- identificar nome de classes e atributos importados;
- definir os tipos dos atributos (string, float, integer, etc.);
- configurar valores mínimo, máximo e intervalo para uma variável de supervisão;
- definir o tempo de atualização de uma variável de supervisão;
- definir o nível de prioridade para cada item de visualização (alto, médio e baixo);

- definir nível de QoS para as visualizações (ouro, prata e bronze);

- selecionar as formas de visualização, através de um catálogo que permite ao projetista selecionar a forma de representatividade desejada para uma variável (imagens, animação, gráfico de barras, texto, etc.);

- configurar o tempo de RTT para o cliente coletar os dados no servidor e exibi-los na tela de supervisão;

- selecionar o local de armazenamento para as formas de visualização, que podem tanto estar no servidor, quanto no cliente. Isso permite que haja uma flexibilidade com os recursos computacionais existentes;

- definir qual o protocolo de comunicação utilizado;

- configurar o endereço da camada de rede *Internet Protocol* (IP) do servidor e o número da porta utilizada para comunicação com o cliente;

- visualizar as ações efetuadas pelo projetista na Ferramenta de Geração.

Figura 3.16 Janela Principal da Ferramenta de Geração

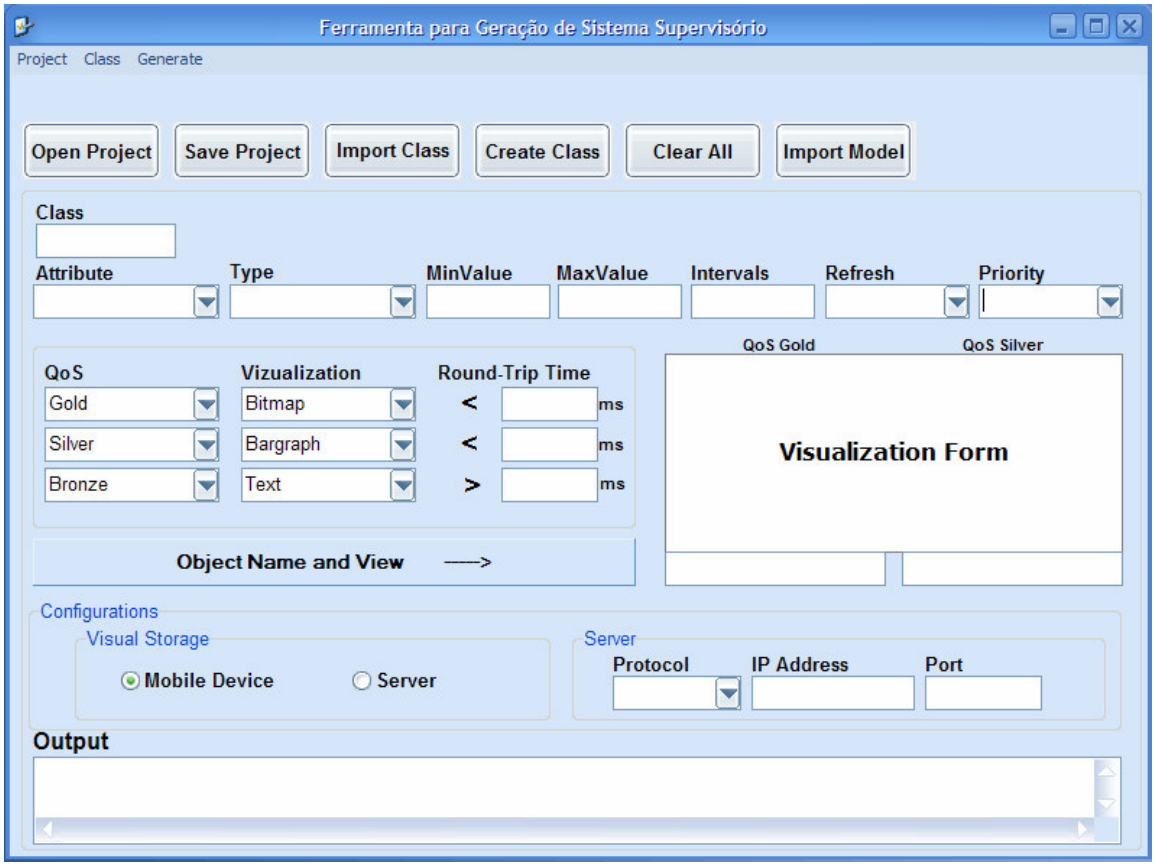

Figura 3.17 Ambiente de Desenvolvimento das Aplicações

A seguir, são apresentados alguns fluxogramas correspondentes ao modo como são implementadas algumas das funcionalidades que permitem à Ferramenta de Geração criar aplicações de supervisão.

A Figura 3.18 apresenta o fluxograma referente ao algoritmo de importação de arquivo XML que contém o modelo orientado a objetos das aplicações. A partir da leitura de um arquivo XML são identificadas classes, atributos e possíveis relacionamentos entre as classes, sendo lidos os elementos que estão aninhados na árvore XML.

Tal função permite identificar o nome dos elementos, o nome dos elementos filhos e os relacionamentos existentes no modelo orientado a objetos da aplicação.

Com essa varredura os campos **Class** e **Attribute** existentes na Ferramenta de Geração recebem os dados lidos do arquivo.

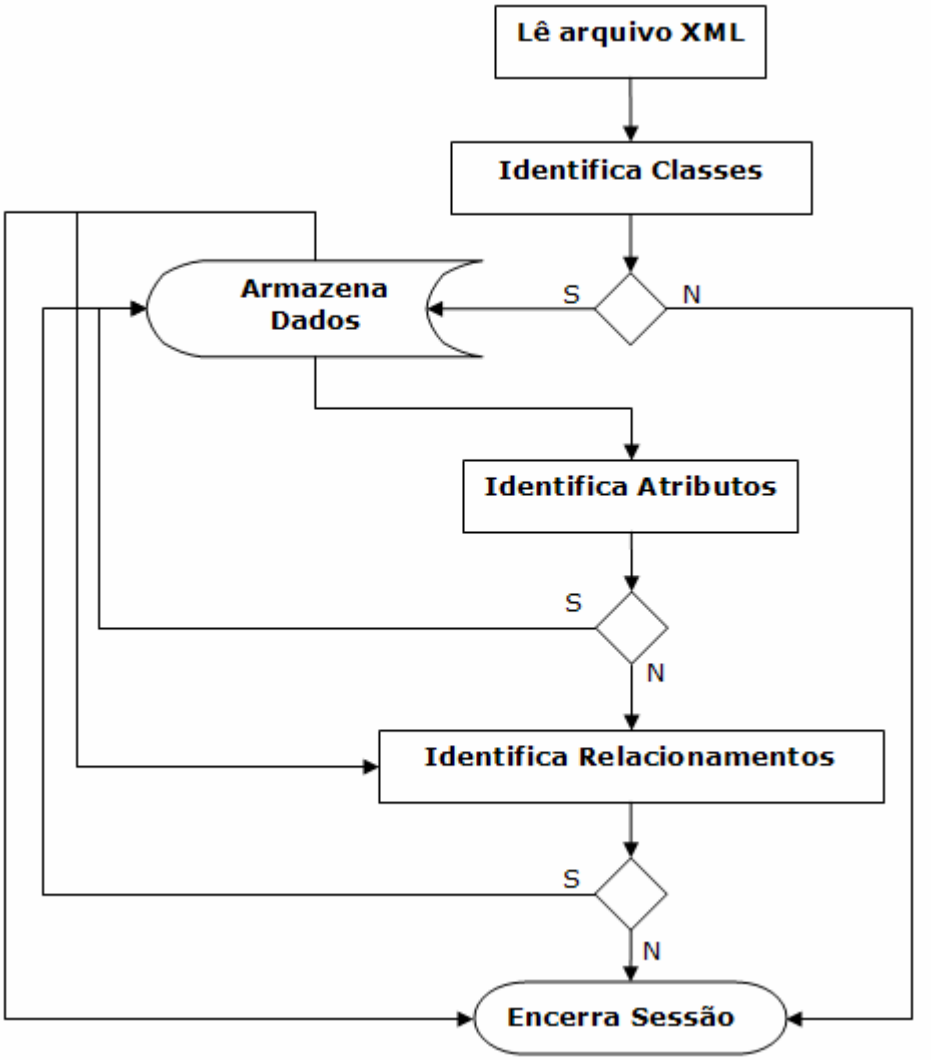

Figura 3.18 Importando classes e atributos de um arquivo XML

A Figura 3.19 apresenta o algoritmo que oferece ao projetista a possibilidade para reuso de projetos já existentes e salvos anteriormente pela Ferramenta de Geração. Essa opção permite que um projeto já criado possa ser editado, reduzindo o tempo de desenvolvimento de uma nova solução.

Após a edição o projetista pode salvar o projeto editado com um novo nome. Esse projeto será adicionado à biblioteca de projetos.

Com a seleção do projeto a ser utilizado, a Ferramenta de Geração percorre o arquivo do projeto aberto em busca dos valores que serão reatribuídos a cada campo existente na janela principal do ambiente de desenvolvimento.

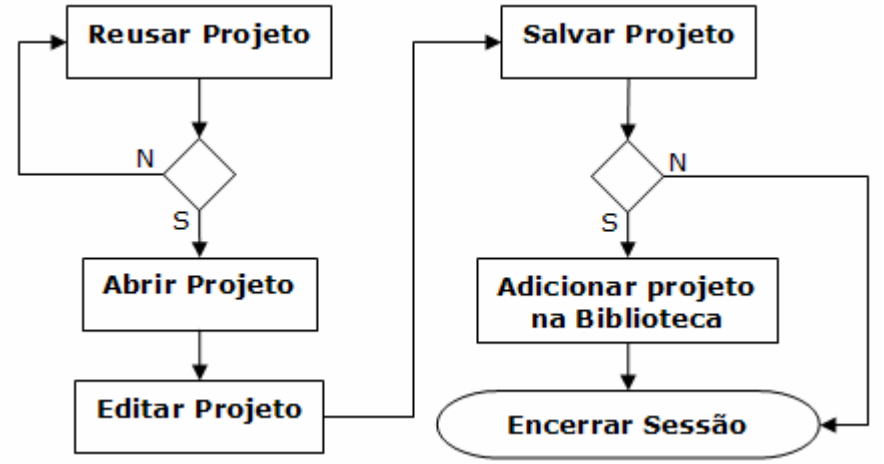

Figura 3.19 Reuso de projetos

A partir da criação de um projeto, a ferramenta possibilita a geração automática de código em linguagem Java. Adicionalmente, o projetista pode optar por salvar ou não o projeto que originou aquele determinado código. Optando por salvá-lo, o projetista pode reutilizá-lo no desenvolvimento de uma outra solução similar ou que necessite de poucas alterações.

As informações referentes aos objetos gráficos são salvas junto com o novo projeto criado, sendo que dois projetos poderiam utilizar um mesmo objeto de visualização com dimensionamentos variados.

Se o projeto de uma aplicação possuir várias instâncias de uma mesma classe, como por exemplo, uma classe tanque, cada instância dessa classe poderia ser representada graficamente de forma genérica ou individual. Ou seja, seria possível atribuir uma forma de visualização diferente para cada instância da classe tanque. O que determina a quantidade de instâncias que uma classe irá ter é o modelo orientado a objetos da aplicação e o que determina as formas de visualização de cada instância é o projetista com base no domínio da aplicação

Além da opção de importação de diagramas de classes descritos em XMI, a ferramenta desenvolvida também possibilita a criação de novas classes através de uma janela com um editor de texto onde o projetista pode editar os nomes das classes e seus atributos, conforme

ilustra a Figura 3.20. Nesse caso não são consideradas as associações entre classes e esta opção apresenta-se como interessante nas situações onde o projetista tenha a necessidade de criar apenas algumas classes individuais ao invés de um modelo orientado a objetos completo.

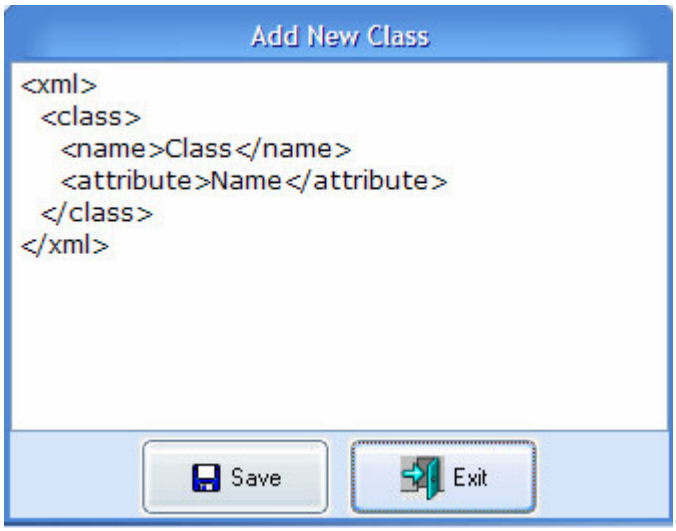

Figura 3.20 Criando Classes e Atributos em XML

A ferramenta, disponibiliza uma barra de menu contendo as opções para geração de código a partir de um projeto criado. Tanto é possível gerar o código Java da aplicação de supervisão para o dispositivo móvel, quanto o arquivo de especificação que é utilizado pelo Coletor de Dados quando este é executado no servidor. Após a geração do código Java, o projetista deverá compilá-lo, através do compilador *Sun Java Wireless Toolkit*, o que resultará de um arquivo com extensão **.java** para outro arquivo com extensão **.jar**, sendo que o último é o arquivo da aplicação de supervisão resultante do projeto desenvolvido e que deverá ser instalado no dispositivo móvel escolhido. Observa-se que a instalação e a execução da aplicação de supervisão gerada está condicionada ao dispositivo móvel que ofereça suporte à linguagem Java e à rede de comunicação de dados.

As informações gráficas que deverão aparecer na tela do sistema supervisório são posicionadas em seqüência, ou seja, uma ao lado da outra até os limites horizontais de cada dispositivo móvel.

A Figura 3.21 apresenta o menu disponível para geração do arquivo com código Java e, em seguida a Figura 3.22 exibe um breve trecho da codificação em linguagem Java que é gerada pela ferramenta.

Figura 3.21 Geração de Código em Java

```
1  import java.io.*;
2  import javax.microedition.midlet.*;
3  import javax.microedition.io.*;
4  import javax.microedition.lcdui.*;
5
6  public class className extends MIDlet {
7    private String urlServer = protocol + IP_Address + port + "datafile.txt";
8    private String addressI =  protocol + IP_Address + port;
9    private Image view = null;
10   private Form form = null;
11   private Display display;
12   private long RTT;
13
14   public className() {
15     display = Display.getDisplay(this);
16   }
17
18 public void startApp() {
19     try {
20       form = new Form("className");
21       getPacket(urlServer);
22     } catch(IOException e) {
23       }
24 }
25 :
```

Figura 3.22 Trecho de código gerado em linguagem Java

Na geração automática de código, a ferramenta gera tanto código estático, de palavras pré-definidas, quanto código dinâmico, inserido pelo projetista na ferramenta em tempo de projeto. Como exemplo de código estático é possível observar a linha 6 da Figura 3.22, através da palavras **public, class, extends e Midlet.** Essas palavras são sempre utilizadas,

independentemente do tipo da aplicação de supervisão que se está projetando. Já para o exemplo de codificação dinâmica, observa-se na mesma linha 6 do mesmo código encontra-se a palavra **className**. Tal palavra refere-se ao nome da classe para a aplicação de supervisão. Esse nome da classe é definido pelo projetista a partir da importação do arquivo XMI, referente ao modelo orientado a objetos da aplicação e que contém os nomes das classes e dos atributos existentes nesse modelo.

No mesmo arquivo que contém o código Java da aplicação de supervisão estão inseridas as codificações responsáveis pelo monitoramento de QoS na rede de comunicação de dados e pela adaptação dinâmica do sistema supervisório em tempo de execução. Esses dois módulos ficam localizados junto à aplicação de supervisão, no lado do cliente, efetuando comunicação e troca de informações, através da rede, com o servidor.

A Figura 3.23 apresenta um diagrama de seqüência da implementação do Monitor de QoS e a Figura 3.24 apresenta um trecho de codificação do Adaptador Online.

Na Figura 3.23 observa-se que o Supervisório ao estabelecer uma conexão, o Monitor de QoS envia e recebe um pacote ao servidor, sendo calculado o RTT. Em seguida o Monitor de QoS solicita os valores correspondentes aos Requisitos de QoS da aplicação e os compara com o valor de RTT obtidos, informando ao Adaptador Online o nível em que se encontra a rede naquele instante. Após, o Supervisório solicita os dados de supervisão ao Servidor, que envia esses dados para o Adaptador Online ajustar a forma de visualização em função da QoS mensurada. Com o ajuste realizado o Adaptador Online devolve ao Supervisório os dados solicitados para a exibição.

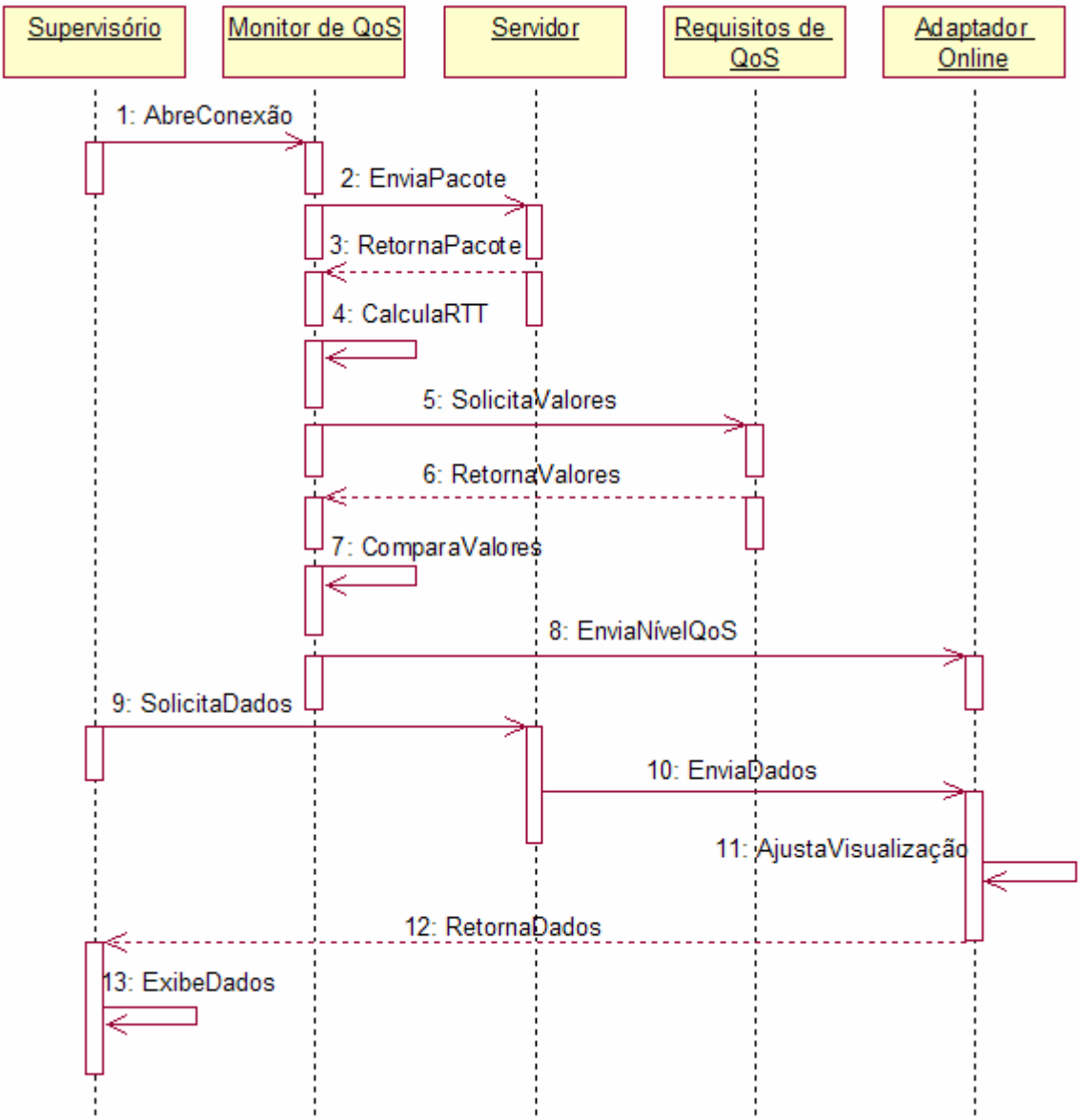

Figura 3.23 Monitor de QoS

```
1   private void OnlineAdaptor(String QoSlevel){
2     if (QoSLevel = "Gold") {
3       view = VisualizationForm("Gold");
4     }
5     if (QoSLevel = "Silver") {
6       view = VisualizationForm("Silver");
7     }
8     if (QoSLevel = "Bronze") {
9       view = VisualizationForm("Bronze");
10    }
11    Form.append(new Item("Supervisory", view, Item.LAYOUT_CENTER, null));
12    Display.setCurrent(Form);
13  }
```

Figura 3.24 Adaptador Online

Quando o projetista define, na Ferramenta de Geração, que as formas de visualização estarão armazenadas no servidor, o Monitor de QoS é utilizado para informar o nível de QoS da rede ao Adaptador Online, para que esse exiba ao usuário as formas de visualização em função da QoS disponível no momento. Outra opção seria a definição automática do Monitor de QoS em função das características do dispositivo móvel escolhido, ou seja, a Ferramenta de Geração armazenaria as configurações de QoS de cada ambiente de execução utilizado em algum projeto. Isso permitiria que a cada novo projeto as informações de QoS daquele *hardware* pudesse ser reaproveitada.

As formas de visualização disponíveis aos usuários poderiam, por exemplo, estar associadas ao tamanho dos dados que estas representam. Ou seja, quanto maior a QoS disponível na rede, mais refinada será a representação visual oferecida ao usuário e, quanto menor essa QoS disponível, mais simples será a forma de apresentação – mas ainda assim contendo as informações mais relevantes. Como exemplo, pode-se considerar que uma medida de QoS com nível "Ouro" resultaria em representação através de animações gráficas, enquanto uma medida de QoS com nível "Bronze", resultaria em informações textuais do processo de supervisão.

Seguindo esse princípio, na Figura 3.24 uma função de adaptação recebe como parâmetro o nível de QoS medido na rede de comunicação de dados pelo Monitor de QoS. Com base nesse parâmetro é chamada uma função que resulta numa forma de visualização correspondente ao nível de QoS mensurado.

3.2.1.2 Coletor de Dados

Essa ferramenta é construída para permitir a troca de dados entre os dispositivos móveis e o servidor. Assim como a Ferramenta de Geração, o Coletor de Dados também é desenvolvida utilizando o ambiente de programação Delphi 7 e a linguagem de programação Object Pascal.

É através do Coletor de Dados que a aplicação de supervisão, instalada no dispositivo móvel, consegue acessar as variáveis de ambiente no servidor e apresentá-las ao usuário. O Coletor de Dados poderia ser implementado para executar tanto no servidor quanto no cliente. Porém, no âmbito deste trabalho se optou por instalá-lo no servidor, realizando a comunicação com um banco de dados que contém valores das variáveis de processo gerados por outros sistemas supervisórios, além de se comunicar diretamente com um sistema de automação. Após coletar as variáveis de processo, o Coletor de Dados gera um arquivo, contendo os dados que foram coletados. Esse arquivo então fica disponível ao cliente, através de um diretório no servidor HTTP. Assim, para que o dispositivo móvel consiga acessar esses dados de supervisão, basta que a *Universal Resource Locator* (URL) do servidor seja conhecida pelo cliente.

Uma questão importante é considerar políticas de segurança para restringir o acesso não autorizado aos dados de supervisão. Em alguns casos apenas evitar a escrita desses dados seria suficiente, mas em outros o procedimento de leitura também deveria ser evitado. Neste trabalho os dispositivos móveis, que executam os sistemas supervisórios, possuem permissão somente para acesso de leitura aos dados, sendo desconsiderada a possibilidade de que os dados no servidor sejam alterados pelos clientes.

Na Figura 3.25 é apresentada a janela principal do Coletor de Dados, onde através das funcionalidades implementas é possível realizar as seguintes tarefas:

- importar arquivo de especificação criado pela Ferramenta de Geração;

- atribuir nome de classe e de atributos contidos no arquivo de especificação;

- especificar o tempo desejado para atualização dos dados de supervisão;

- selecionar em qual diretório do servidor HTTP se deseja gerar o arquivo de dados para o cliente;

- selecionar qual a banco de dados será utilizado para coletar os valores das variáveis de supervisão;

- definir o endereço do servidor onde está o banco de dados para acesso;

- inserir o nome do banco de dados a ser utilizado;

- informar o nome e a senha do usuário que tem acesso autorizado ao banco de dados;

- listar dos dados de supervisão coletados e disponibilizados ao cliente.

Figura 3.25 Coletor de Dados

Inicialmente, a ferramenta Coletor de Dados permite que seja importado o arquivo de especificação criado pela Ferramenta de Geração. É realizada uma varredura no arquivo de especificação e os dados lidos são atribuídos aos campos do Coletor de Dados, por exemplo: a primeira linha do arquivo de especificação contém o nome da classe, sendo atribuído ao campo **ClassName**. Também poderia ser implementado um código para verificação de consistência entre cliente e servidor, garantindo que ambos estivessem utilizando o mesmo arquivo de especificação.

Após realizada a importação dos dados de especificação, é possível selecionar o local de saída onde ficará o arquivo contendo os dados de supervisão coletados na base de dados ou no sistema de automação. Esse arquivo de saída deverá ser salvo dentro de um diretório do servidor, estando disponível para acesso do cliente via protocolo HTTP.

Identificados o nome da classe e dos atributos, através da importação do arquivo de especificação e definidos o tempo de atualização dos dados juntamente com o local de saída destes, são inseridas no Coletor de Dados as informações para acesso ao banco de dados, tais como: endereço, nome, usuário e senha. Com essas informações conhecidas, a Figura 3.26 apresenta o procedimento de como os dados de supervisão são coletados no banco de dados. É realizada uma consulta na tabela com o nome da classe, retornando os valores dos atributos contidos nessa classe.

```
procedure TForm1.ConnectDataBase(Sender: TObject);
begin
  sqlquery1.Active := false;
  sqlquery1.SQL.Clear;
  sqlquery1.SQL.Add('SELECT * FROM ' + EditClassName.Text);
  sqlquery1.ExecSQL;
  sqlquery1.open;
  sqlquery1.First;
  begin
    while not sqlquery1.Eof do
      begin
        EditValor1.Text:=sqlquery1.
                  FieldByName(EditAttribute1.Text).AsString;
        sqlquery1.Next;
      end;
  end;
```

Figura 3.26 Coletar o valor das variáveis no banco de dados

Na Figura 3.26 são apresentados alguns comandos em linguagem *Structured Query Language* (SQL) adotada neste trabalho pelo de ser uma linguagem padroniza de pesquisa e manipulação de banco de dados relacional, estando presente nos mais variados sistemas de informação e podendo ser encontrada em simples aplicativos que são executados em PDAs ou em consultas complexas realizadas em *mainframes*.

A utilização da SQL permite que a aplicação responsável pela leitura dos dados de supervisão na planta industrial possa realizar consultas em diferentes bases de dados, obtendo nome de tabelas e valores de atributos que correspondam às variáveis do processo de supervisão que se deseje monitorar.

Com os valores dos atributos conhecidos, o Coletor de Dados executa um procedimento para salvar esses valores, atualizando o arquivo de saída que corresponde aos dados de supervisão que o cliente terá acesso, sendo que essa atualização é realizada a cada intervalo de tempo definido pela opção de *Refresh*.

3.2.2 Software Utilizados

- Para o desenvolvimento do trabalho são utilizados, além do sistema operacional: Microsoft Windows XP – Service Pack 2 (MICROSOFT, 2007):

▪ **Rational Rose Enterprise Edition** - Versão 2003.06.15.734.000: ferramenta de modelagem orientada a objetos com suporte a geração de arquivos XMI (IBM, 2006);

▪ **Borland Delphi 7**: ambiente de desenvolvimento integrado com edição de código através da linguagem Object Pascal (BORLAND, 2007);

▪ **Sun Java Wireless Toolkit** (WTK) - Versão 2.3: kit de compilação para aplicações J2ME (SUN WTK, 2007) e **JDK** - Versão 5: kit de desenvolvimento Java (JAVA SE, 2007).

- Para execução das aplicações de supervisão geradas faz-se necessário:

▪ **IBM WebSphere Everyplace Micro Environment** – Versão 6.1: máquina Virtual Java para rodar aplicações J2ME em dispositivos móveis baseados no sistema operacional Windows Mobile 5.0 (IBM J9, 2007);

▪ **Apache Web Server** - Versão 2.2.4: servidor HTTP (APACHE, 2007);

▪ **MySQL –** Versão 4.1: banco de dados que utiliza a linguagem SQL (MYSQL, 2007).

4 VALIDAÇÃO DO TRABALHO

Este capítulo apresenta a validação do trabalho proposto através de três estudos de casos, sendo que no primeiro há uma demonstração, passo a passo, de como são utilizadas as ferramentas implementadas neste trabalho, bem como as demais ferramentas que o compõe.

4.1 ESTUDO DE CASO 1: ELIPSE SCADA

Este estudo de caso demonstra como é possível a integração entre o sistema supervisório Elipse SCADA e os sistemas supervisórios para dispositivos móveis projetados no âmbito deste trabalho. A principal motivação deste estudo é permitir a conectividade entre as aplicações de supervisão projetadas com outros sistemas supervisórios disponíveis no mercado.

4.1.1 Definição do Processo de Supervisão

O processo de supervisão escolhido neste estudo de caso consiste em supervisionar o nível de um tanque, presente numa aplicação do sistema supervisório Elipse SCADA. O Elipse SCADA armazena o valor atualizado da variável supervisionada num banco de dados, sendo acessado pelo Coletor de Dados e após disponibilizado o valor da variável para o cliente.

4.1.2 Modelagem Orientada a Objetos da Aplicação

Definido o processo de supervisão, cria-se o modelo orientado a objetos da aplicação, através do diagrama de classes em UML, que é construído com a utilização da ferramenta de modelagem Rational Rose. Após a construção do modelo orientado a objetos deve-se gerar, na própria ferramenta Rational Rose, o arquivo XMI correspondente ao modelo, conforme ilustra a Figura 4.1.

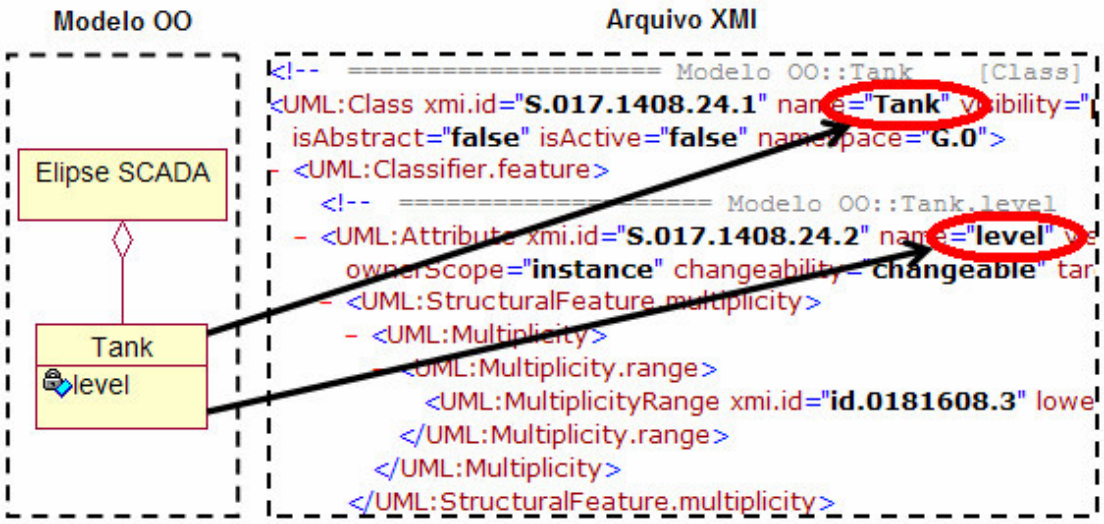

Figura 4.1 Modelo Orientado a Objetos e Arquivo XMI Correspondente

4.1.3 Projeto da Aplicação de Supervisão

Com a modelagem e a geração do arquivo XMI, a aplicação de supervisão é projetada através da Ferramenta de Geração, na qual são configurados todos os itens que resultam no sistema supervisório.

A primeira tarefa a ser realizada pelo projetista na Ferramenta de Geração é a importação do modelo orientado a objetos da aplicação, que, nesse caso, é importado o arquivo XMI desse modelo, conforme ilustra a Figura 4.2.

Após o modelo ter sido importado, são definidas pelo projetista as propriedades da variável de supervisão, tais como: a faixa de operação, o tempo para atualização, as prioridades, o intervalo entre valores mínimo e máximo, as formas de visualização associadas aos possíveis níveis de QoS encontrados na rede de comunicação de dados, a definição dos valores de RTT para cada nível, a especificação do protocolo de comunicação utilizado e o local para armazenamento do catálogo de visualizações que, neste estudo de caso, encontra-se no servidor.

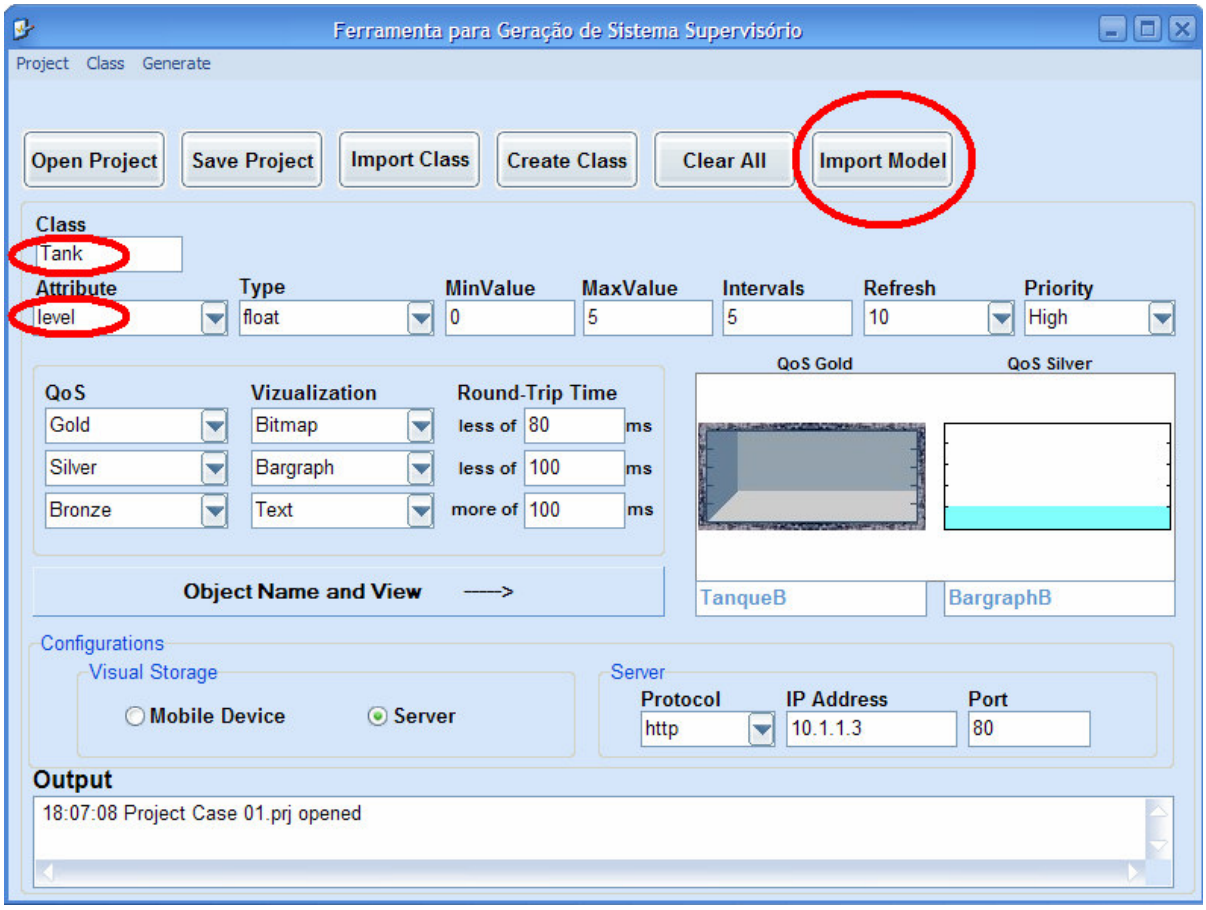

Figura 4.2 Importação do Modelo Orientado a Objetos

Com todos os campos da Ferramenta de Geração preenchidos, o projeto é salvo na biblioteca de projetos, sendo gerados o arquivo de especificação para o Coletor de Dados que roda no servidor (Figura 4.3) e o código Java da aplicação (Figura 4.4).

```
- <xml>
  - <specification>
      <Class>Tank</Class>
      <Attribute>level</Attribute>
      <Type>float</Type>
      <MinValue>0</MinValue>
      <MaxValue>5</MaxValue>
      <Intervals>5</Intervals>
      :
      :
```

Figura 4.3 Especificação do Coletor de Dados

```
1  import java.io.*;
2  import javax.microedition.midlet.*;
3  import javax.microedition.io.*;
4  import javax.microedition.lcdui.*;
5
6  public class Tank extends MIDlet {
7
8    private Display display;
9    private String url = "http://10.1.1.3:80/datafile.txt";
10   private Form formulario = null;
11   private Image img = null;
12   private String body;
13
14   public Tank() {
15     display = Display.getDisplay(this);
16   }
17   :
     :
```

Figura 4.4 Trecho do Código Java da Aplicação

A partir do código Java gerado para a aplicação de supervisão, este é compilado pela ferramenta *Sun Java Wireless Toolkit* que, por sua vez, é iniciada, sendo criado um novo projeto, conforme apresenta a Figura 4.5.

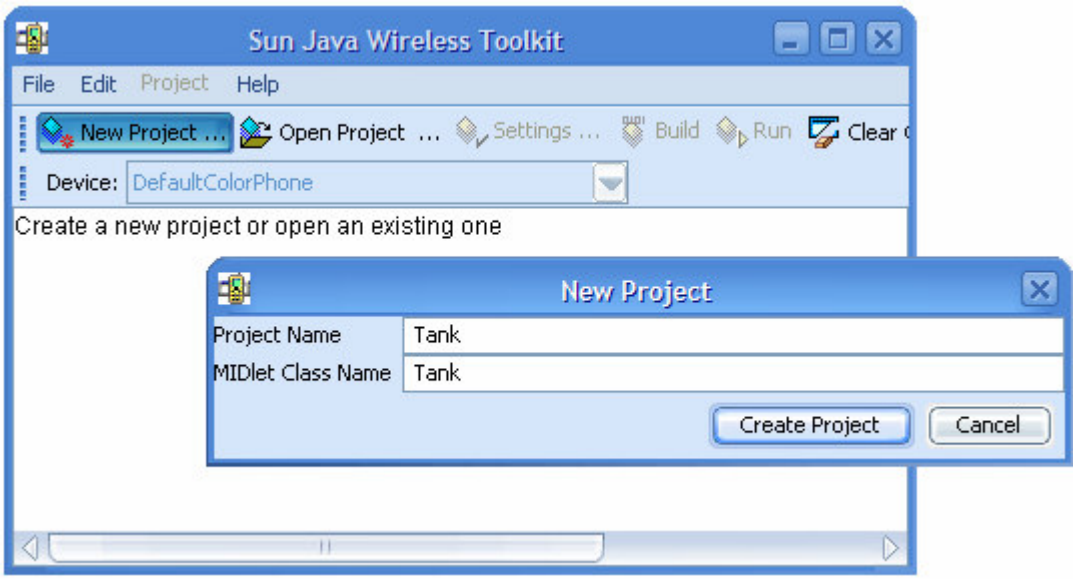

Figura 4.5 Novo Projeto na Ferramenta Sun Java Wireless Toolkit

O nome do projeto, criado no Toolkit da Sun, tem o mesmo nome da classe projetada. Após criado o projeto, um novo diretório chamado Tank, é adicionado ao diretório raiz do

Toolkit. Esse diretório Tank possui, por padrão, outros subdiretórios descritos de acordo com a Tabela 2:

Tabela 2 Subdiretórios do projeto criado

Subdiretório	**Conteúdo**
bin	Bytecodes (arquivos .jad e .jar)
lib	Bibliotecas extras
res	Imagens utilizadas
src	Código fonte da aplicação

O arquivo contendo o código Java, gerado pela Ferramenta de Geração, é copiado pelo projetista para o subdiretório **src**, juntamente com as imagens referentes catálogo de visualizações que também são copiadas para o subdiretório **res**.

O projeto é compilado e gerado um pacote pelo Toolkit da Sun, resultando nos arquivos Tank.jad e Tank.jar disponibilizados no subdiretório **bin**.

O arquivo Tank.jar criado é instalado no dispositivo móvel que irá realizar o processo de supervisão.

Instalada a aplicação de supervisão (Tank.jar) no dispositivo móvel, é utilizada a ferramenta Coletor de Dados para ler os dados gerados pelo Elipse SCADA, disponibilizando-os no Servidor HTTP para o acesso do dispositivo móvel.

Com o Coletor de Dados em execução no servidor, é importado o arquivo de especificação, sendo identificada a classe e a variável de processo a ser supervisionada, conforme ilustra a Figura 4.6.

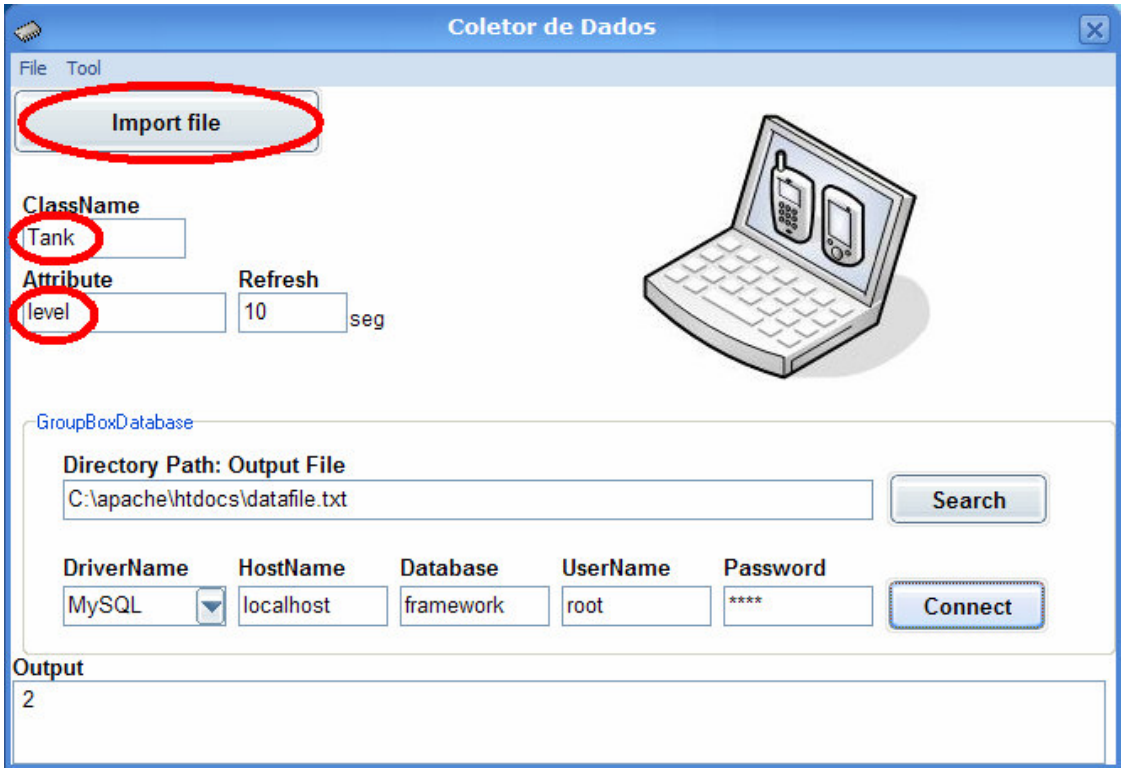

Figura 4.6 Coletor de Dados em Execução

Em seguida, são definidos a base de dados utilizada pelo Elipse SCADA, o endereço dessa base de dados, o nome e a senha para acesso aos dados. Então o Coletor de Dados realiza a leitura do valor da variável no banco de dados, disponibilizando esses valores em um arquivo no diretório definido para saída dos dados e que é acessado pelo cliente.

4.1.4 Supervisão através do Emulador de Dispositivos Móveis

Os primeiros testes para a supervisão do processo definido neste estudo de caso são realizados através da emulação de um dispositivo móvel. A emulação é realizada através do próprio emulador de aplicações existentes no Toolkit da Sun.

Após todo o desenvolvimento do projeto da aplicação de supervisão, é no emulador que essa aplicação é validada pela primeira vez, pois a mesma aplicação que rodar no emulador poderá executar em dispositivos móveis reais que ofereçam suporte à linguagem Java.

Desse modo, são estabelecidos para testes os seguintes ambientes e condições:

▪ **Teste 1**: Emulação da aplicação de supervisão gerada, através de uma rede *Local Area Network* (LAN) sem carga. O dispositivo móvel emulado num computador, roda a aplicação de supervisão e acessa os dados gerados pelo sistema supervisório Elipse SCADA, conforme ilustra a Figura 4.7.

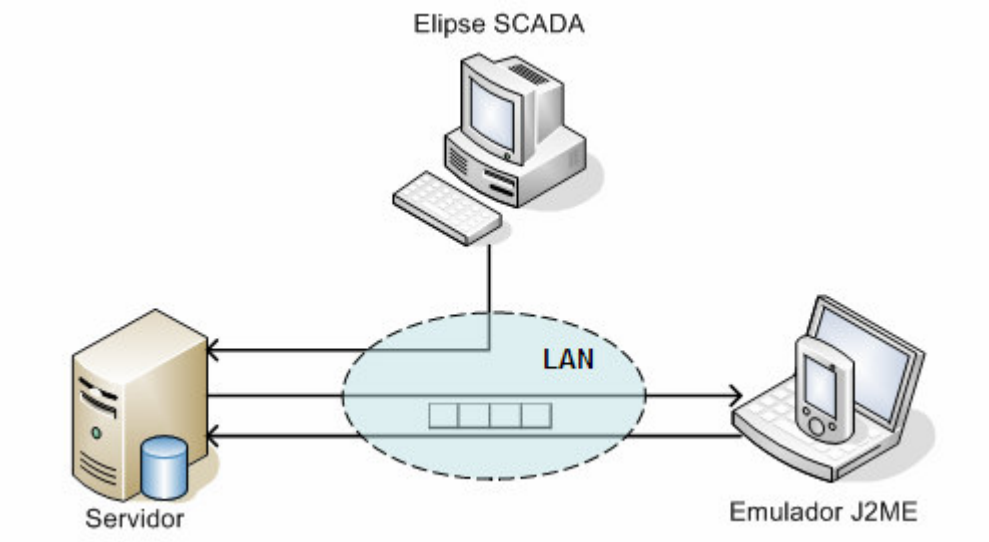

Figura 4.7 Cenário do Teste 1

As configurações de *hardware* e *software* utilizadas neste Teste 1 foram: computadores Servidor, Emulador J2ME e Elipse SCADA, os quais possuem sistema operacional Microsoft Windows XP Profesisonal, Service Pack 2, processador AMD Athlon XP2800+ 2.26GHz, 512 MB de RAM. A rede utilizada para a comunicação de dados é uma LAN, com tecnologia Ethernet de 10Mbps.

Como a visualização dos dados é dada em função da QoS disponível na rede de comunicação, foram coletados os valores de RTT entre cliente e servidor, sem carga na rede, ou seja, quando não há outros computadores utilizando a rede de comunicação além dos computadores Servidor, Emulador J2ME e Elipse SCADA.

A Figura 4.8 apresenta o gráfico com os valores de RTT medidos para um pacote de 2 KBytes.

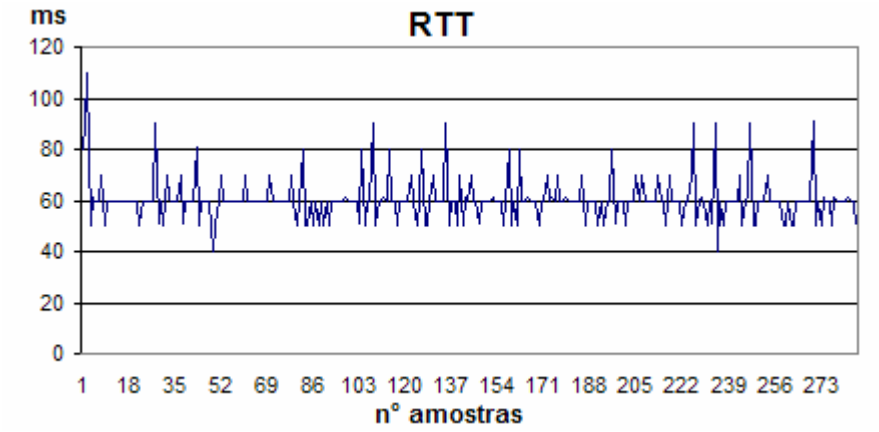

Figura 4.8 Gráfico do RTT sem carga na rede

A partir dos valores obtidos, foi verificado um valor médio de 60 ms para o RTT quando não há carga na rede.

▪ **Teste 2**: Emulação da aplicação de supervisão gerada, através de uma rede LAN com carga. O dispositivo móvel emulado num computador, a exemplo do Teste 1, também roda a aplicação de supervisão e acessa os dados gerados pelo sistema supervisório Elipse SCADA, só que dessa vez um outro computador utiliza a rede de comunicação de dados realizando solicitações ao servidor, conforme ilustra a Figura 4.9.

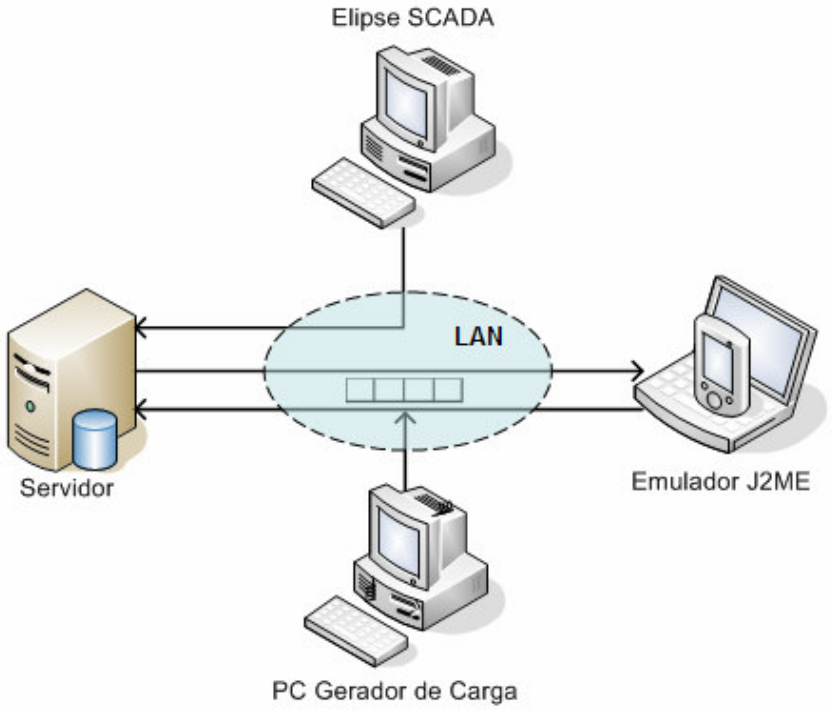

Figura 4.9 Cenário do Teste 2

As configurações de *hardware* e *software* utilizadas no Teste 2 são as mesmas definidas no Teste 1, adicionado apenas o computador Gerador de Carga. A rede utilizada para a comunicação de dados também é a mesma do Teste 1.

Para o Teste 2 foram coletados os valores de RTT entre cliente e servidor, com carga na rede, ou seja, com a presença de um outro computador responsável por gerar tráfego na rede de comunicação de dados. Essa geração de tráfego foi realizada através do computador Gerador de Carga que gerava constantemente solicitações ao computador servidor, através do comando *ping Servidor -t -l 65500*, realizado no *prompt* de comando. A Figura 4.10 apresenta o gráfico com os valores de RTT medidos nessas condições.

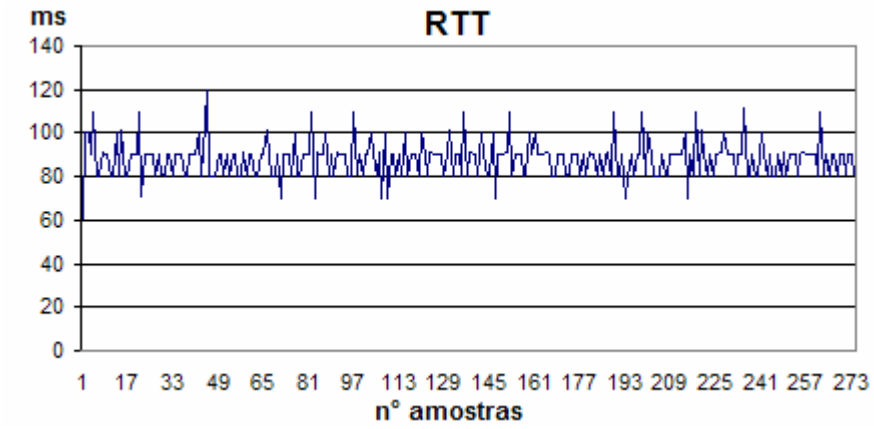

Figura 4.10 Gráfico do RTT com carga na rede

Desse modo, com a geração de carga na rede de comunicação, o valor médio de RTT ficou estabelecido em 87,82 ms.

- **Teste 3**: Emulação da aplicação de supervisão gerada através da Internet. O dispositivo móvel emulado num computador roda a aplicação de supervisão e acessa via Internet os dados que são gerados pelo sistema supervisório Elipse SCADA. Para o Teste 3 definiu-se que os computadores Servidor e Elipse SCADA estariam na mesma rede LAN, sendo que o Servidor poderia ser acessado via Internet, conforme ilustra a Figura 4.11.

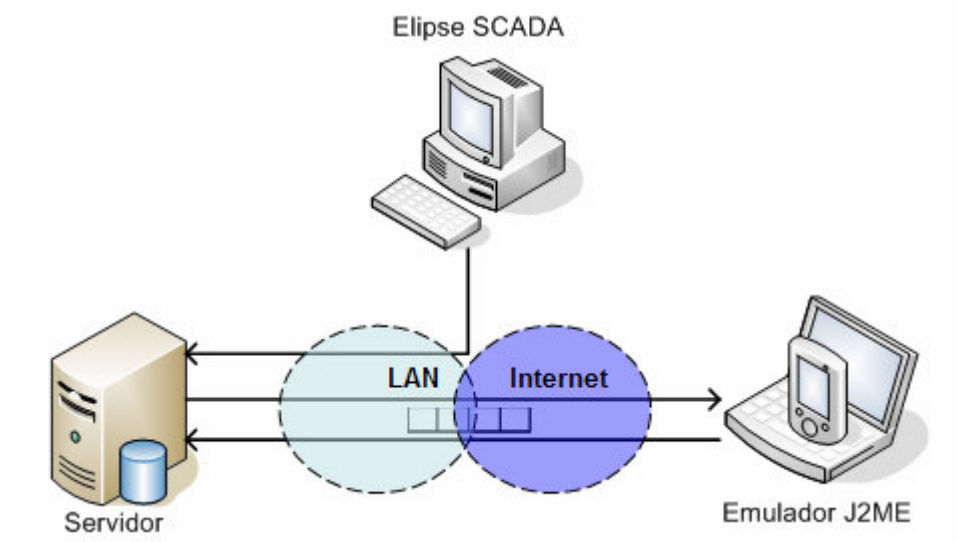

Figura 4.11 Cenário do Teste 3

As configurações de *hardware* e *software* utilizadas no Teste 3 permanecem as mesmas definidas no Teste 1. Inclui-se ao Teste 1 apenas a Internet como sendo o alvo das medidas de QoS.

Com a finalidade de medir a QoS disponível da comunicação de dados entre cliente e servidor via Internet, foram coletados novos valores de RTT, conforme o gráfico apresentado na Figura 4.12, sendo que a média de RTT mensurada nesse teste foi de 162,01 ms.

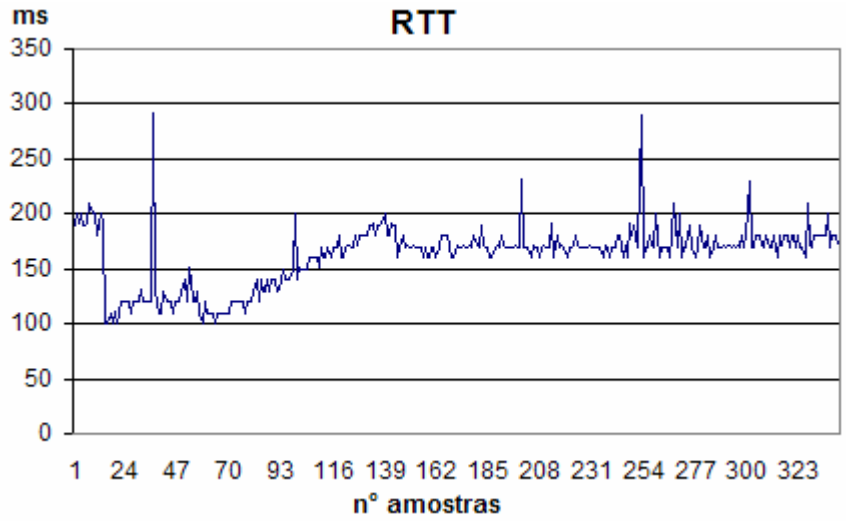

Figura 4.12 Gráfico do RTT via Internet

O Teste 3 serviu para verificar como a QoS utilizando a Internet sofreu uma variação mais significativa se comparada aos testes anteriores. Ao final dos Testes 1 e 2 é possível

analisar os valores de RTT coletados na mesma rede LAN, em diferentes condições, sendo plotados lado a lado, conforme o gráfico exibido na Figura 4.13. Percebe-se que na mesma rede LAN existem variações de QoS quando há e quando não há carga na rede.

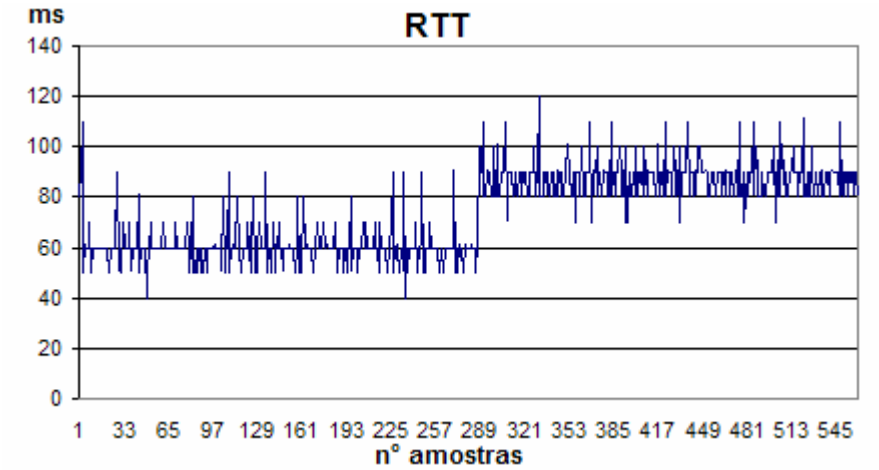

Figura 4.13 Gráfico do RTT Com e Sem Carga na Rede

A primeira metade do gráfico corresponde aos valores de RTT coletados sem carga na rede; já a segunda metade do gráfico, representa os valores de RTT coletados com carga na rede. É baseado nessas medidas que são definidos os níveis de QoS da rede de comunicação, conforme Figura 4.14.

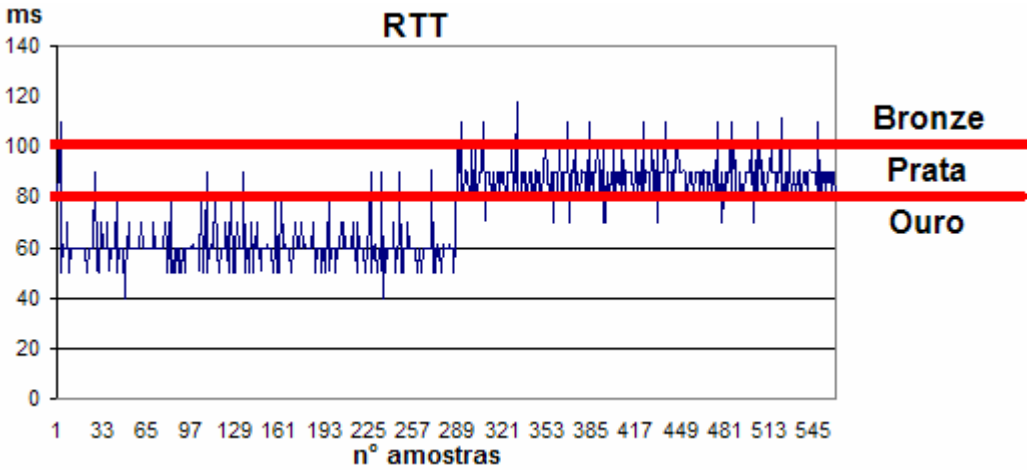

Figura 4.14 Gráfico com os Níveis de QoS

Assim, a aplicação de supervisão considera a QoS disponível na rede como sendo:

Ouro: para valores de RTT menores ou iguais a 80ms;

Prata: para valores de RTT entre 80 e 100 ms;

Bronze: para valores de RTT superiores a 100 ms.

Para cada nível de QoS uma distinta forma de visualização, definida pelo projetista na Ferramenta de Geração, seria utilizada.

Assim, a supervisão do processo através do Emulador, resultou em momentos de alternância no modo como apresentou os dados na tela do dispositivo móvel, conforme apresenta a Figura 4.15.

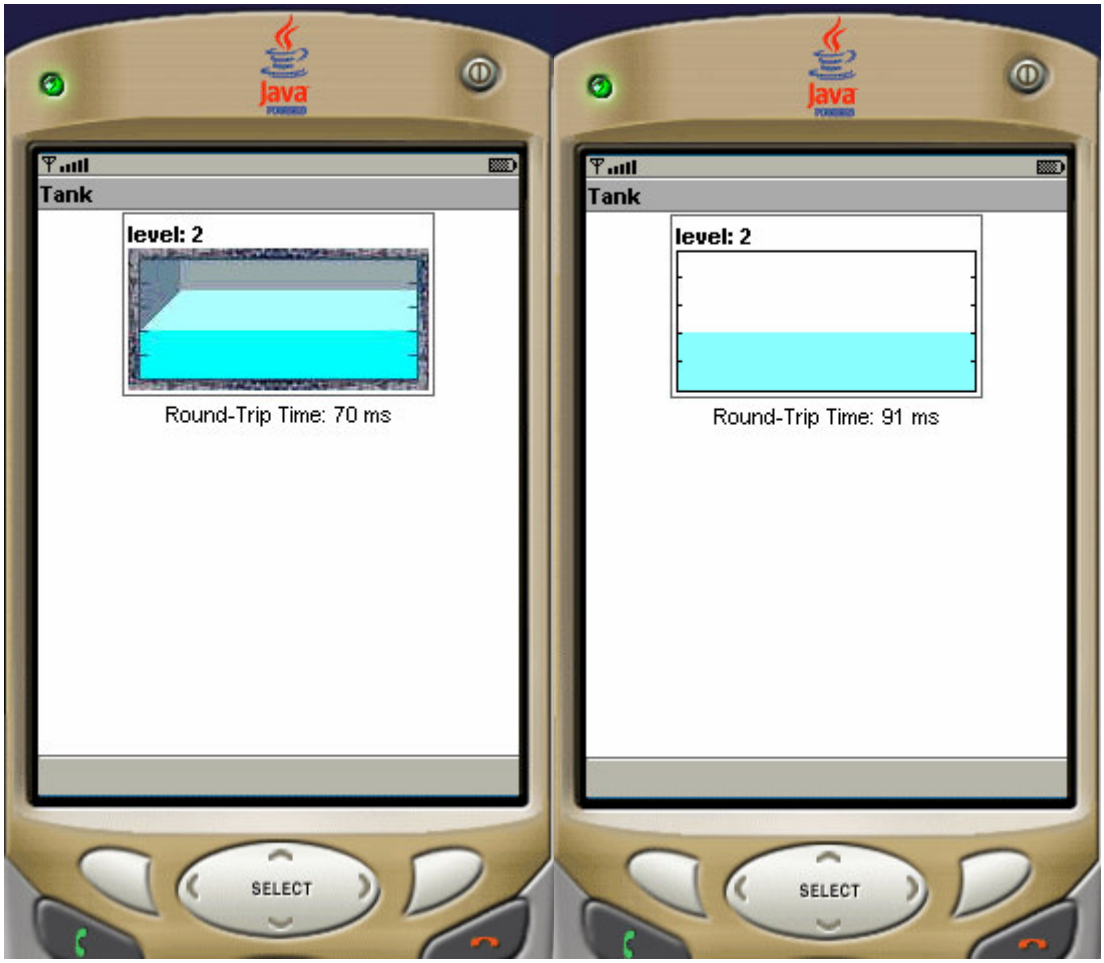

Figura 4.15 Tela de Supervisão Emulada

Quando o nível de QoS da rede era “Ouro” o usuário tinha disponível uma forma de visualização com mais qualidade e definição do processo. Todavia, quando o nível de QoS da rede era “Prata” o usuário teve disponível um bargraph, uma visualização com qualidade intermediária. Por fim, quando o nível de QoS da rede foi raramente “Bronze”, restou disponibilizado ao usuário apenas a informação textual do processo de supervisão.

4.1.5 Supervisão Através de um Dispositivo Móvel Real

Após realização dos testes iniciais para a aplicação de supervisão terem sido realizados através do emulador existente no Toolkit da Sun, definiu-se um novo ambiente de testes, similar aos testes anteriores. Porém, retirando o computador responsável pela emulação e adicionando um dispositivo móvel real. A mesma aplicação de supervisão criada pela Ferramenta de Geração e utilizada pelo emulador nos testes anteriores é utilizada nesses novos testes, os quais possuem as seguintes condições:

- **Teste 4:** Após instalada a aplicação de supervisão ela é executada no dispositivo móvel que, por sua vez, utiliza uma rede local WLAN para comunicação com o Servidor. A rede LAN para comunicação de dados utilizada entre os computadores Servidor e Elipse SCADA se mantém como nos testes anteriores, sendo adicionado ao novo cenário uma WLAN de 11Mbps, conforme ilustra a Figura 4.16.

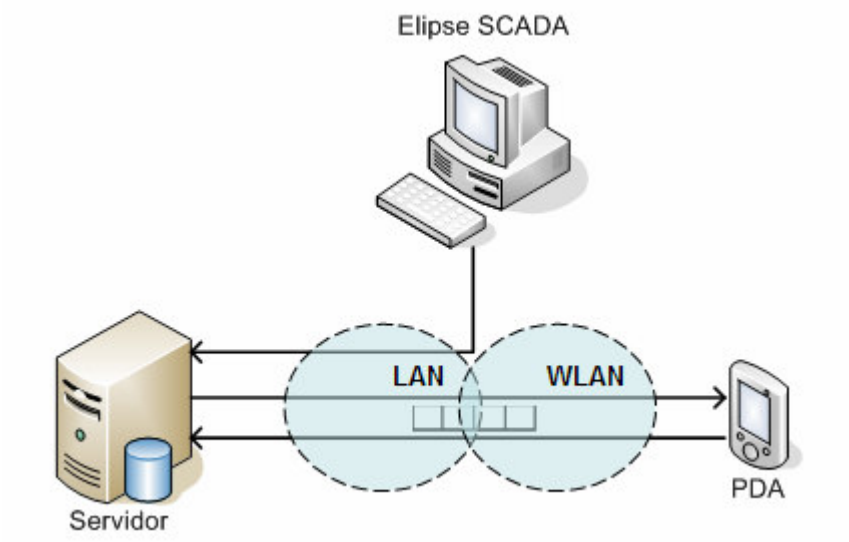

Figura 4.16 Cenário do Teste 4

As configurações de *hardware* e *software* utilizadas no Teste 4 foram: computadores Servidor e Elipse SCADA possuem sistema operacional Microsoft Windows XP Profesisonal, Service Pack 2, processador AMD Athlon XP2800+ 2.26GHz, 512 MB de RAM. O dispositivo móvel escolhido é um Dell Axim X51v, processador Intel XScale PXA270 de

624MHz, 64MB de RAM, com sistema operacional Microsoft Windows Mobile 5.0 e com a JVM J9 da IBM.

Inicialmente, é verificado a QoS disponível nas redes utilizadas para comunicação de dados entre o PDA e o Servidor. Nessa verificação foram coletados valores de RTT bem superiores aos valores coletados inicialmente através da emulação, conforme o gráfico que pode ser observado na Figura 4.17.

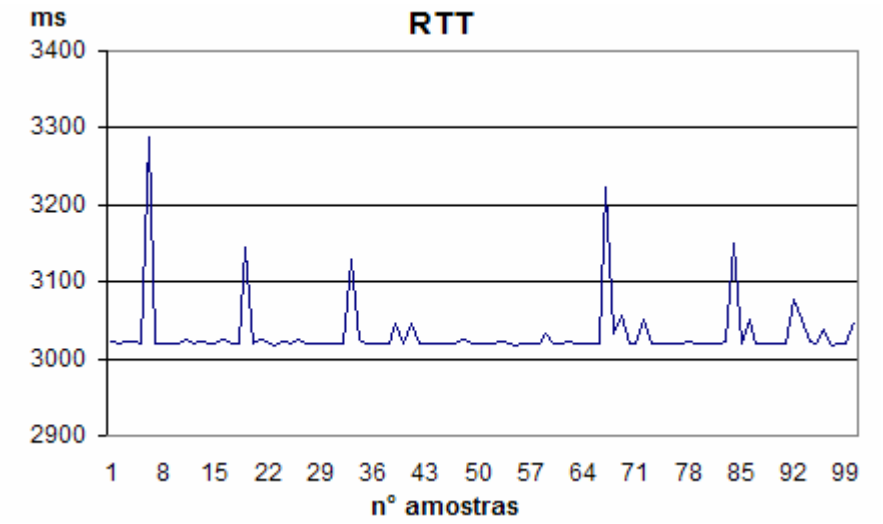

Figura 4.17 Gráfico do RTT utilizando PDA numa WLAN

A média resultante dessas amostras ficou em 3031,87 ms, sendo que esses valores ficaram altos se comparados com os testes anteriores nos quais o computador executou o emulador em condições semelhantes.

▪ **Teste 5:** Esse teste segue o mesmo princípio do Teste 4, sendo adicionada ao cenário a rede Internet como forma do PDA coletar os dados de supervisão no servidor, conforme ilustra a Figura 4.18.

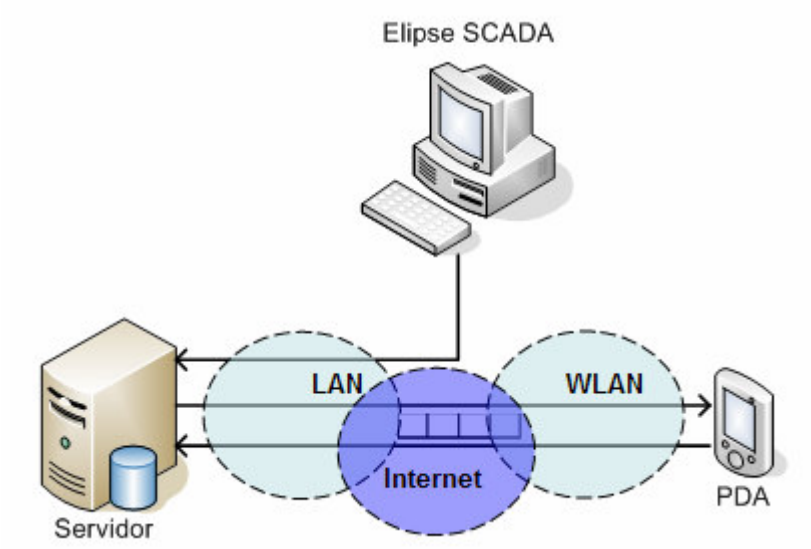

Figura 4.18 Cenário do Teste 5

As configurações de *hardware* e *software* utilizadas nesse teste também seguem as mesmas utilizadas no teste anterior.

Desse modo, verificou-se novamente a QoS disponível nas redes utilizadas para comunicação de dados entre o PDA e o Servidor, resultando, como era esperado, numa coleta de valores de RTT superiores aos valores obtidos no Teste 4, conforme o gráfico apresentado na Figura 4.19

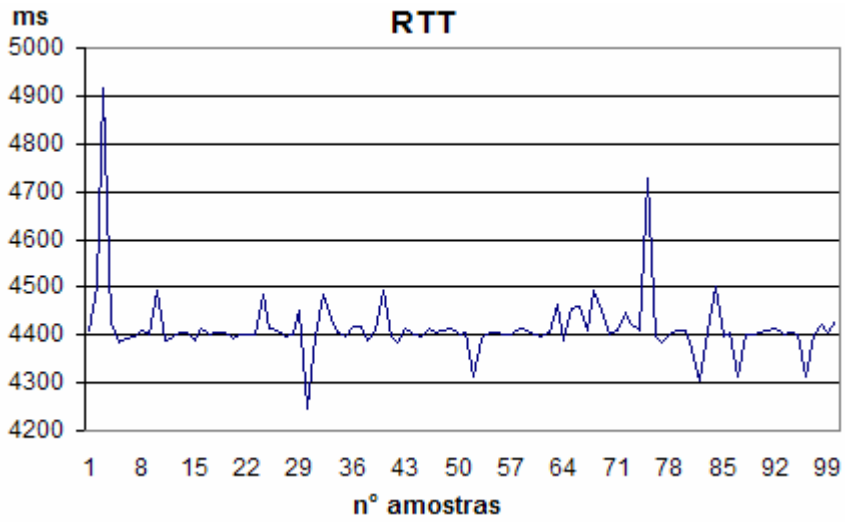

Figura 4.19 Gráfico do RTT utilizando PDA e Internet

A média dos valores de RTT amostrados foi de 4416,16 ms, ficando em torno de 31,34% acima da média obtida no teste anterior.

Tanto o Teste 4 quanto o Teste 5, serviram para definir um novo valor para os níveis de QoS existentes nas redes de comunicação de dados utilizadas. Ou seja, foi necessário definir um novo valor de RTT para cada nível de QoS definido e para cada cenário criado, por exemplo:

No Teste 4, foi assumida a QoS disponível na rede de comunicação como sendo:

Ouro: para valores de RTT menores ou iguais a 3100 ms;

Prata: para valores de RTT entre 3100 e 3200 ms;

Bronze: para valores de RTT superiores a 3200 ms.

Já para o Teste 5, a QoS disponível na rede de comunicação foi assumida como:

Ouro: para valores de RTT menores ou iguais a 4500 ms;

Prata: para valores de RTT entre 4500 e 4800 ms;

Bronze: para valores de RTT superiores a 4800 ms.

A aplicação de supervisão criada neste estudo de caso foi testada em cinco cenários diferentes. Nos três primeiros testes a aplicação de supervisão foi emulada e nos dois testes seguintes, a mesma aplicação foi instalada e executada num PDA, realizando a supervisão através de um dispositivo móvel real.

Foram definidos vários níveis de QoS para cada cenário de testes, cujas formas de visualização estavam diretamente associadas ao nível de QoS disponível na rede de comunicação de dados.

Como resultado final deste estudo de caso, é possível observar na Figura 4.20 a integração entre a aplicação de supervisão criada e o sistema supervisório Elipse SCADA.

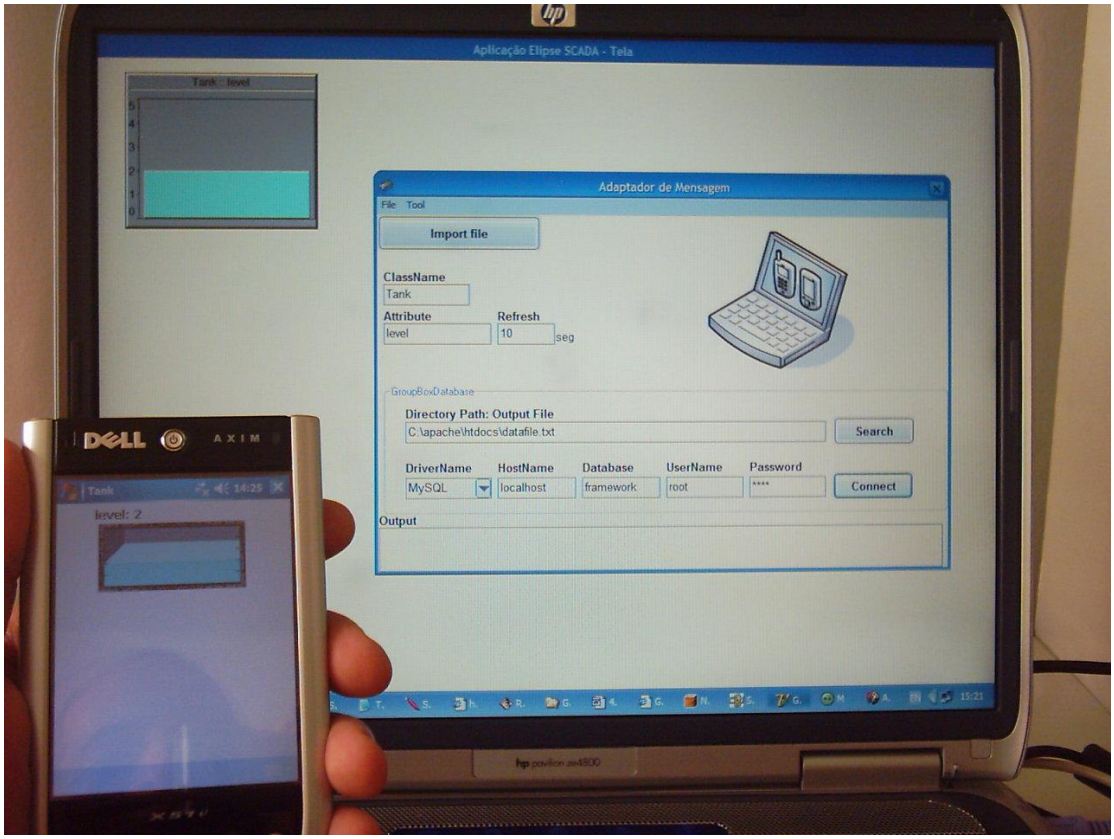

Figura 4.20 Resultado da Integração com Elipse SCADA

4.2 Estudo de Caso 2: Sistema de Automação Predial/Residencial da empresa HomeSystems

A Homesystems é uma empresa de automação predial/residencial (HOMESYSTEMS, 2007), gaúcha, com sede em Porto Alegre. A empresa desenvolve projetos de pesquisa aplicada em cooperação com o Grupo de Controle, Automação e Robótica (GCAR) desta universidade.

O conjunto de soluções projetado pela empresa Homesystems consiste numa arquitetura com diferentes módulos conectados em rede. Esses módulos possuem tanto funcionalidades independentes quanto funcionalidades dependentes de um módulo controlador central chamado *Systembox*. O *Systembox* é, basicamente, um computador com sistema operacional Linux, responsável por controlar a rede *HomeSystems Network* (HSNET - protocolo proprietário que opera sobre a camada física RS-485) e executar funcionalidades,

oferecendo comunicação remota com todos os módulos da arquitetura, através de uma interface *web*. A arquitetura da Homesystems disponibiliza, ainda, um conjunto de dispositivos que podem ser utilizados em sistemas de iluminação, ar condicionado e sistemas de segurança.

4.2.1 Definição do Cenário de Supervisão

Dentre todos os módulos disponíveis na arquitetura da Homesystems, foi utilizado para este estudo de caso os módulos *Systembox* e *Quicklight*, sendo este último responsável pelo sistema de iluminação, com o qual é possível definir cenários baseados na intensidade desejada para a iluminação de um determinado ambiente.

Assim, o cenário de supervisão escolhido baseia-se nos equipamentos para testes disponibilizados pela empresa Homesystems ao GCAR, conforme apresenta a Figura 4.21. Especificamente é supervisionado o estado das quatro lâmpadas presentes no módulo *Quicklight*, sendo que esta supervisão é realizada através da comunicação entre as aplicações projetadas por este trabalho com o *Systembox*, responsável pelo gerenciamento central da arquitetura Homesystems. O estado das quatro lâmpadas são representados pelo valor zero ou um, sendo que o valor zero indica lâmpada desligada e o valor um indica lâmpada ligada.

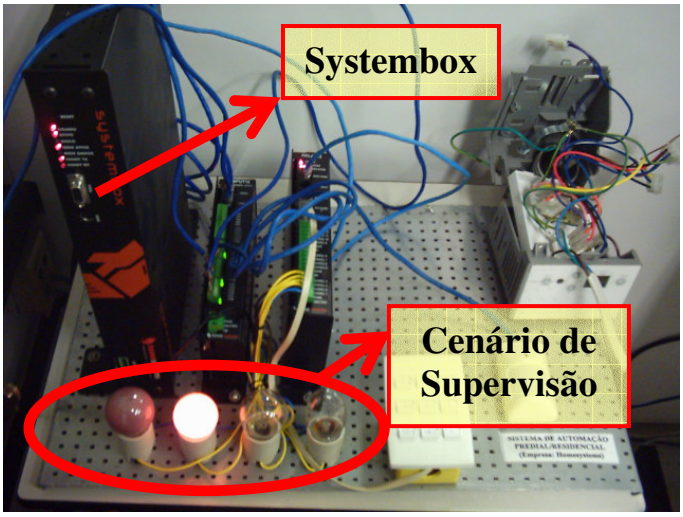

Figura 4.21 Cenário de Supervisão Homesystems

Já na Figura 4.22 são apresentados o modelo orientado a objetos e o arquivo XMI criados, que representam computacionalmente o cenário de supervisão do mundo real utilizados neste estudo caso e ilustrado na Figura 4.21.

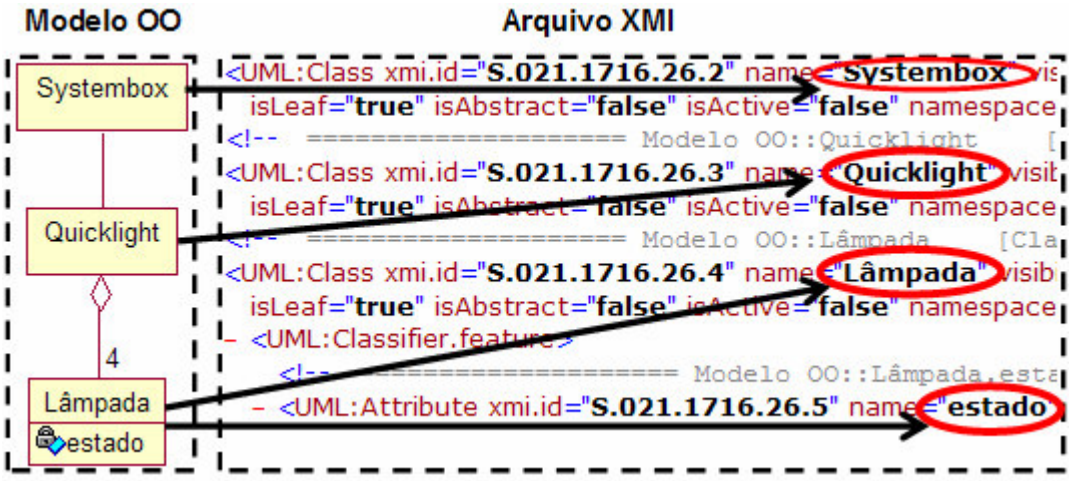

Figura 4.22 Modelo Orientado a Objetos e Arquivo XMI

4.2.2 Supervisão com PDA

Após a aplicação de supervisão ter sido projetada seguindo todos os passos de utilização das ferramentas exibidos no primeiro estudo de caso, é iniciado um novo teste de validação do trabalho proposto.

Desse modo, é estabelecido para o teste o seguinte ambiente e condição:

▪ **Teste**: A aplicação de supervisão é instalada e executada no dispositivo móvel, que se comunica com o servidor através de uma rede local WLAN 11Mbps. A comunicação entre o servidor e o *Systembox* é realizada através de uma rede LAN 10Mbps. A Figura 4.23 apresenta o cenário da comunicação de dados.

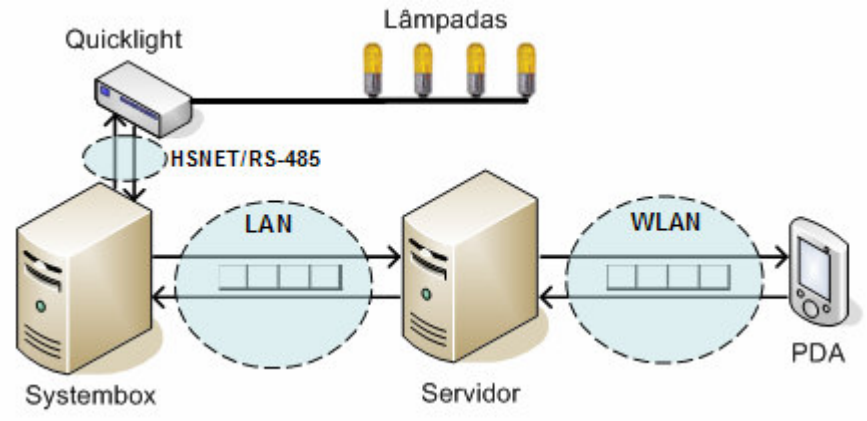

Figura 4.23 Cenário da Comunicação de Dados

As configurações de *hardware* e *software* utilizadas no Teste foram: computador Servidor possui sistema operacional Microsoft Windows XP Profesisonal, Service Pack 2, processador AMD Athlon XP2800+ 2.26GHz, 512 MB de RAM. O PDA é o mesmo utilizado nos estudo de caso anterior, sendo um Dell Axim X51v, processador Intel XScale PXA270 de 624MHz, 64MB de RAM, com sistema operacional Microsoft Windows Mobile 5.0 e com a JVM J9 da IBM.

No estudo de caso anterior, o Coletor de Dados realizava a leitura das variáveis num banco de dados, disponibilizando-os para acesso dos dispositivos móveis. Já para esse estudo de caso, o Coletor de Dados foi implementado de forma a acessar diretamente o controlador central ao invés de buscar os dados num banco de dados, acessando diretamente o controlador central de automação da empresa Homesystems, o *Systembox*, que oferece suporte de acesso através do protocolo HTTP.

As implementações realizadas no coletor de Dados são importantes para identificar as possibilidades para o acesso às variáveis de supervisão como, por exemplo, as implementações realizadas que acessam banco de dados e sistemas controladores de dispositivos de automação.

Novas formas de acesso podem ser implementadas e adicionadas ao Coletor de Dados. Uma possibilidade seria a comunicação de dados entre cliente e servidor através do protocolo OPC.

A exemplo do estudo de caso anterior, foi considerada a QoS da rede de comunicação de dados, a qual influencia diretamente as formas de visualização e adaptações dos dados em tempo de execução do sistema supervisório projetado.

Assim foram coletados os valores de RTT na comunicação entre cliente e servidor, conforme o gráfico apresentado na Figura 4.24.

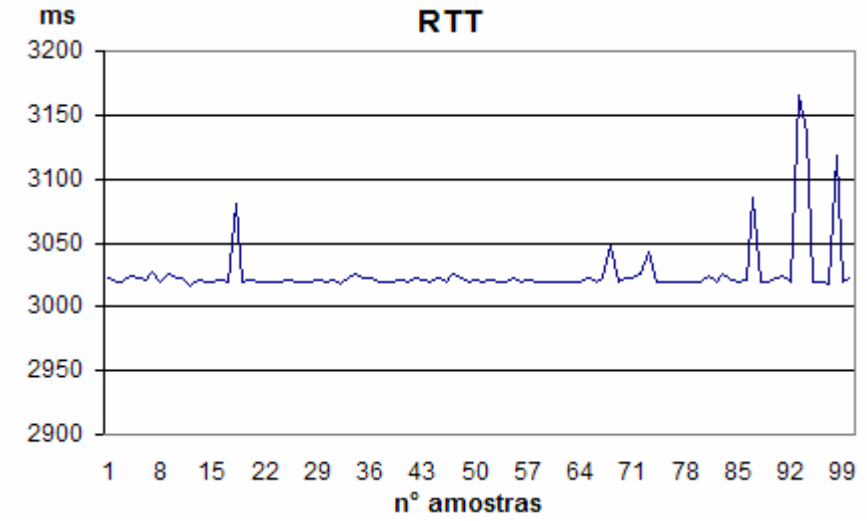

Figura 4.24 QoS em função do RTT

Com a leitura dos valores, foi obtido um RTT médio de 3026,01 ms, sendo assumidos esses valores obtidos para atribuição dos níveis de QoS. Com a identificação dos níveis de QoS encontrados na rede de comunicação de dados, foi estabelecido que a QoS da rede estaria com os seguintes níveis: **Ouro**, quando os valores de RTT não ultrapassarem 3050 ms; **Prata**, para valores de RTT entre 3050 e 3100ms; e **Bronze**, para valores de RTT superiores a 3100ms, conforme a Figura 4.25. Baseado nesses níveis foi possível definir as formas de visualização em tempo de execução, o que resultou na supervisão do módulo *Quicklight* controlado pelo *Systembox*, conforme apresenta a Figura 4.26.

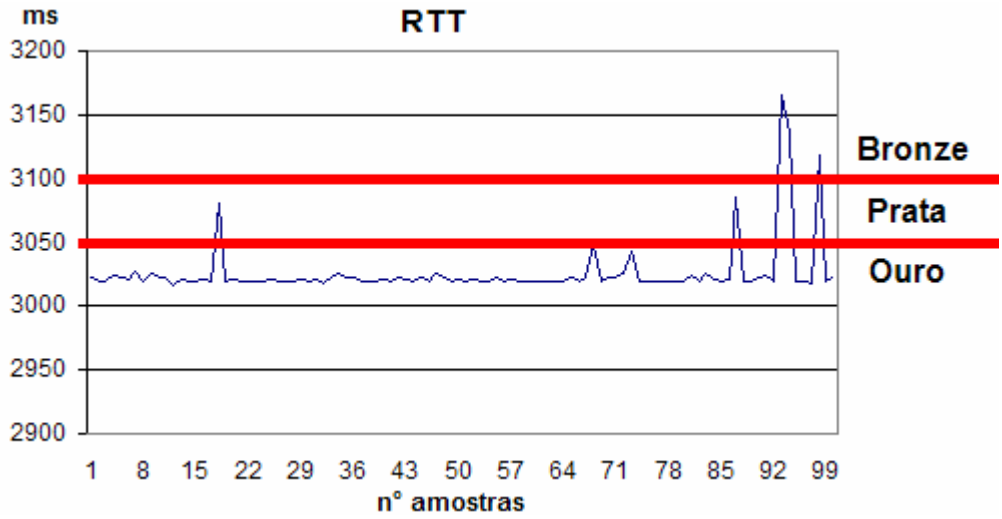

Figura 4.25 Nível de QoS atribuído aos valores de RTT

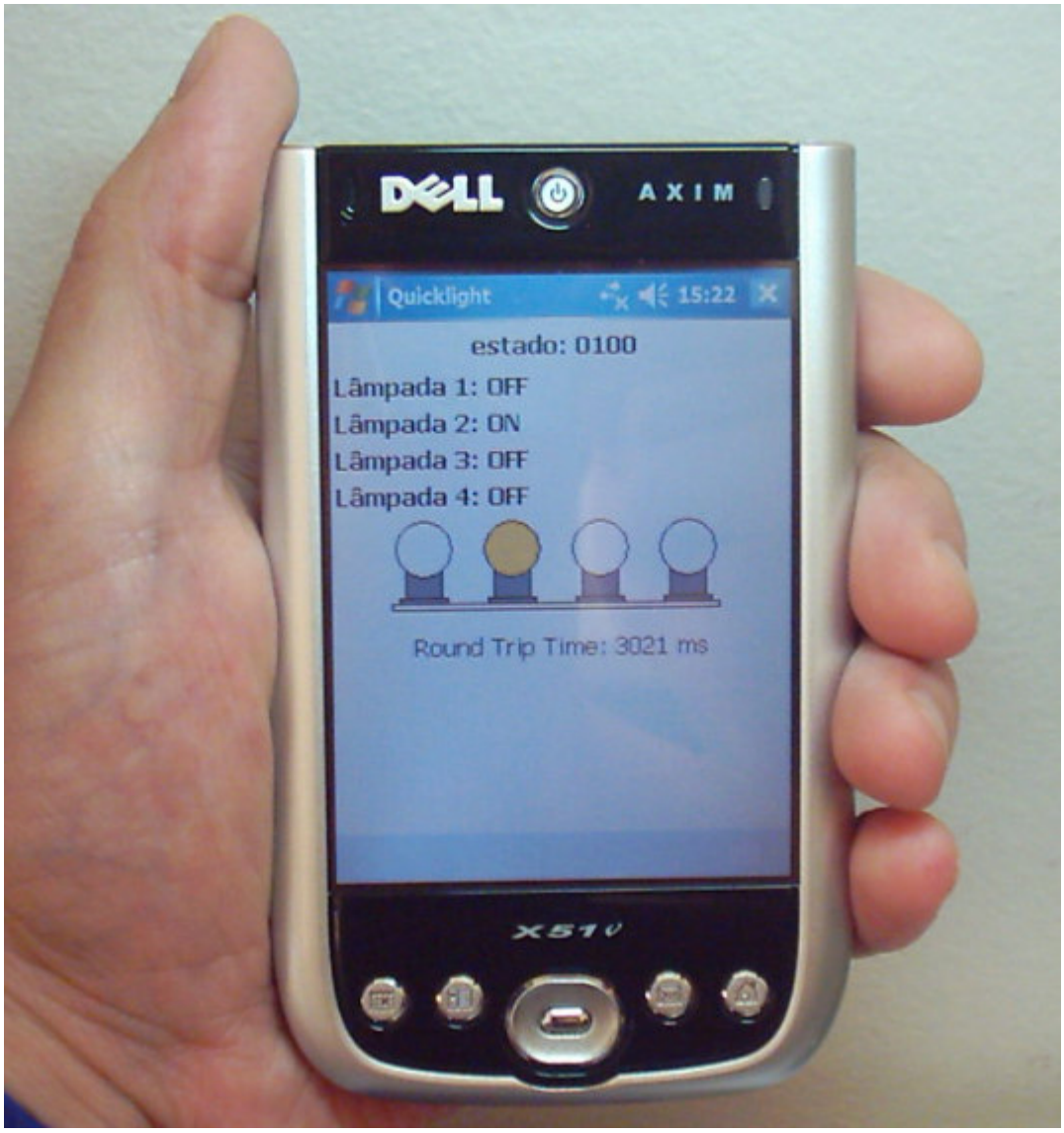

Figura 4.26 Supervisão do módulo Quicklight

4.3 Estudo de Caso 3: Supervisão de um Simulador de Processos

Este estudo de caso é motivado pela possibilidade de testar o comportamento da aplicação de supervisão quando existe mais de um dispositivo de automação para ser monitorado. Nesse caso um dispositivo pode ter requisitos de QoS, tempo de atualização e prioridade de visualização diferente de outro.

4.3.1 Definição da Supervisão

O processo de supervisão escolhido neste estudo de caso consiste na supervisão da temperatura e da quantidade de litros atual em três tanques. Para isso foi utilizado um *software* desenvolvido para ser um simulador de processos, sendo responsável por gerar valores aleatórios da quantidade atual de litros e da temperatura de cada tanque, conforme pode ser observado na Figura 4.27.

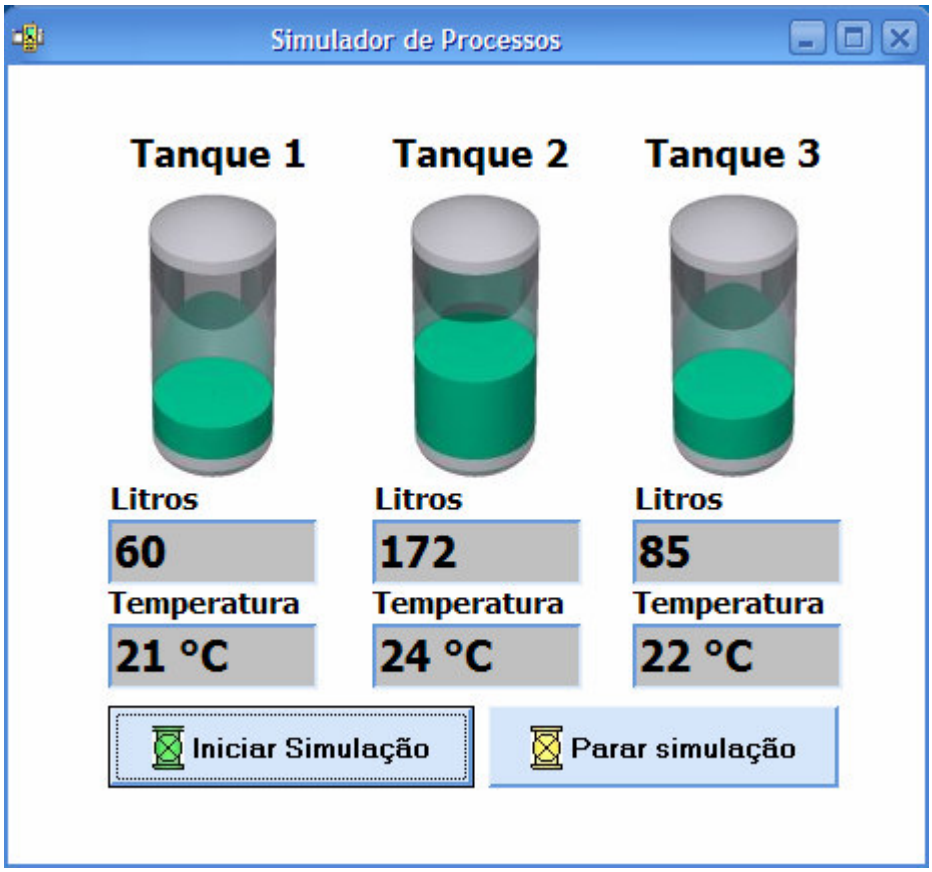

Figura 4.27 *Software* para a Simulação de Processos

4.3.2 Projeto de Supervisão

Para verificar o comportamento do sistema de supervisão mediante o monitoramento de mais de um dispositivo de automação, foram definidos alguns requisitos para cada dispositivo:

Para primeiro tanque o valor da variável temperatura deveria ser atualizado a cada cinco segundos e o valor da quantidade de litros seria atualizado a cada quatro segundos. A prioridade de visualização assumida para esse tanque em relação aos outros seria alta, ou seja, o primeiro tranque é mais importante para visualização que os demais.

No segundo tanque definiu-se que o valor da variável temperatura deveria ser atualizado a cada seis segundos e para a quantidade de litros a cada cinco segundos. Foi assumida para o segundo tanque uma prioridade média de visualização.

Para o terceiro e último tanque a ser supervisionado foi definido que a temperatura seria monitorada a cada cinco segundos e a quantidade de litros a cada três segundos. Já a visualização foi assumida como de baixa prioridade.

4.3.3 Supervisão do Processo

Após a aplicação de supervisão ter sido projetada considerando os requisitos estabelecidos, é iniciado o teste para este estudo de caso.

Assim, o ambiente de testes possui as seguintes condições:

A aplicação de supervisão é instalada e executada no dispositivo móvel, que se comunica com o servidor através de uma rede local WLAN 11Mbps. No servidor é executado o Simulador de Processos onde este disponibiliza os dados de simulação gerados para a supervisão através de uma rede LAN 10Mbps. A Figura 4.28 apresenta o cenário da comunicação de dados.

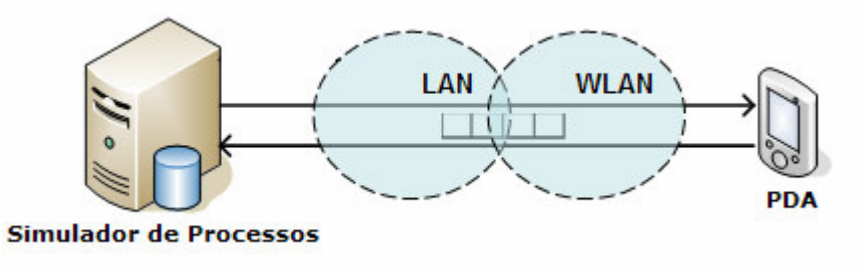

Figura 4.28 Cenário da Simulação e Supervisão

As configurações de *hardware* e *software* utilizadas nesse Teste foram: computador Servidor possui sistema operacional Microsoft Windows XP Profesisonal, Service Pack 2, processador AMD Athlon XP2800+ 2.26GHz, 512 MB de RAM. O PDA é o mesmo utilizado nos estudos de casos anteriores.

Para cada dispositivo de automação foram assumidos requisitos diferenciados, sendo desconhecido neste caso a QoS da rede antes da execução do sistema supervisório. Ou seja, a idéia é verificar o comportamento do sistema supervisório em tempo de execução, a partir das configurações estabelecidas, mesmo quando há uma variação na QoS da rede de comunicação de dados. Assim, a Figura 4.29 apresenta o gráfico com os valores de RTT medidos para um pacote de 2 KBytes.

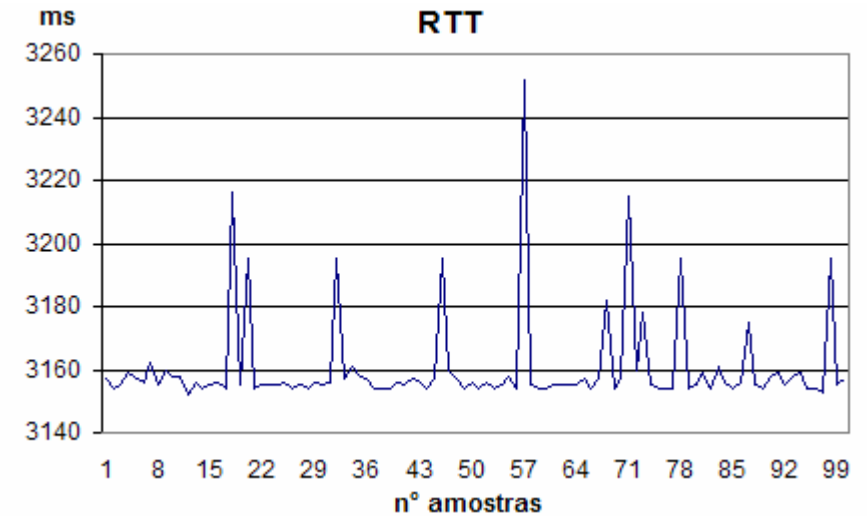

Figura 4.29 Gráfico dos valores de RTT obtidos

Nesse teste de comunicação foi mensurado um RTT médio de 3160,58 ms, sendo que com esses valores obtidos os requisitos especificados em tempo e projeto puderam ser

parcialmente atendidos em tempo de execução, pois o menor valor para atualização das variáveis de supervisão era de três segundos.

Uma visualização do sistema supervisório sendo executado pode ser observada na Figura 4.30.

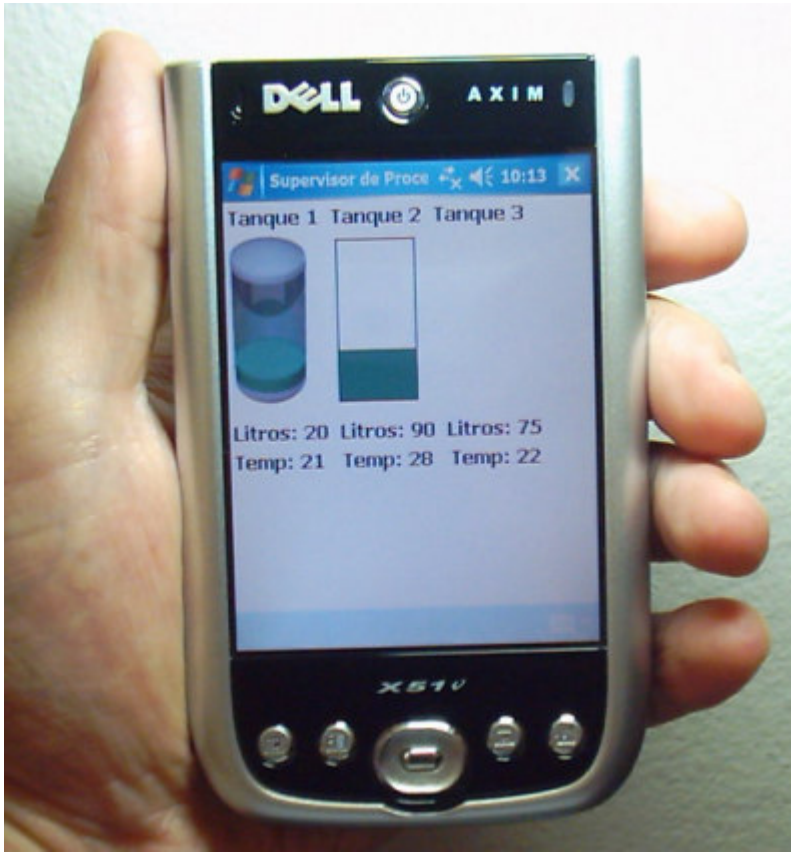

Figura 4.30 Sistema supervisório em execução

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi proposto um *framework* que permite o desenvolvimento de sistemas supervisórios para dispositivos móveis.

Para tanto, reuniu-se um conjunto de temas, na análise do estado da arte, tendo sido identificados vários desafios científicos e tecnológicos que serviram de apoio para transformar simples dispositivos móveis em plataformas de supervisão.

A utilização do paradigma de orientação a objetos serviu como base para a construção das telas de supervisão, pois através dos modelos orientados a objetos foram mapeados os conceitos existentes no mundo real para os conceitos computacionais. Além disso, as técnicas de orientação a objetos permitem a reutilização de projetos criados anteriormente, acelerando o tempo no desenvolvimento de uma nova solução em condições similares.

Com a definição e a implementação de uma arquitetura, foi possível desenvolver sistemas supervisórios para o domínio da aplicação escolhido. As técnicas implementadas permitiram que os sistemas supervisórios gerados pela Ferramenta de Geração oferecessem flexibilidade em relação aos recursos de *hardware* existentes nos dispositivos móveis, que em tempo de projeto era possível definir se o armazenamento dos dados multimídia que representam os gráficos de supervisão estariam armazenados no servidor ou no próprio dispositivo móvel. A técnica de adaptabilidade em tempo de execução permitiu que as telas de supervisão fossem ajustadas dinamicamente de acordo com a qualidade e a disponibilidade dos serviços oferecidos pela infra-estrutura de comunicação de dados, sendo que a QoS era monitorada, pelo dispositivo móvel, através da técnica de RTT, medindo o tempo de ida e volta de uma mensagem entre cliente e servidor. Esse tempo de RTT obtido em tempo de execução era comparado com os valores de RTT definidos em tempo de projeto, resultando na atribuição de um nível de QoS (ouro, prata e bronze) para cada valor de RTT medido. Adicionalmente, a cada nível de QoS existia uma forma de visualização associada, o que

permitia que o operador visualizasse os dados em função de uma métrica temporal pré-definida e da QoS atual da rede.

Foi desenvolvida, também, uma ferramenta que coleta os dados de supervisão, em banco de dados ou em sistemas controladores, disponibilizando-os para os dispositivos móveis através de um servidor HTTP. Isso permitiu verificar, nos três estudos de caso, que é totalmente possível integrar as aplicações de supervisão que são geradas por este trabalho com outros *softwares* de supervisão e sistemas de controle disponíveis no mercado. Ainda nos estudos de caso foi possível verificar que mesmo quando os níveis de QoS degradam, através do mecanismo de ajuste automático, o sistema consegue se adaptar, apresentando ao operador informações significativas do processo de supervisão, sejam estas textuais ou gráficas. Baseado nesse contexto, o trabalho proposto oferece a possibilidade de serem criados sistemas supervisórios para dispositivos móveis atendendo a um mercado em potencial, utilizando técnicas para adaptabilidade e flexibilidade de recursos computacionais, considerando o *hardware* do dispositivo móvel, os *softwares* de supervisão e as redes de comunicação de dados.

Como trabalho futuro, destaca-se: (i) a implementação de uma comunicação OPC para as aplicações de supervisão projetadas, visando habilitar a comunicação com diversos dispositivos de automação e sistemas de monitoramento, independentemente do fabricante ou do *software*; (ii) acrescentar um módulo para definição dinâmica das formas de visualização, dispensando a necessidade de se criar uma série de visualizações estáticas que representam graficamente uma ação; (iii) verificar as possibilidades tanto de segurança quanto da viabilidade em ser adicionado ao trabalho um módulo para acionamento de comandos através dos dispositivos móveis; e (iv) incluir a QoS no ambiente de execução dos sistemas supervisórios.

Outras validações deste trabalho estão sendo realizadas por um acadêmico no âmbito de conclusão do curso de Ciência da Computação nas Faculdades Integradas Facvest, em Lages-SC.

5.1 PUBLICAÇÕES

Este trabalho resultou na publicação de quatro artigos científicos:

▪ PEROZZO, Reiner F.; PEREIRA, C. E. "Framework for Building Supervisory Systems in Mobile Devices". In: 11th IEEE International Conference on Emerging Technologies and Factory Automation - ETFA 2006, 2006, Praga. Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation - ETFA 2006, 2006. p. 167-172.

▪ PEROZZO, Reiner F.; PEREIRA, C. E. "Framework para Construção de Sistemas Supervisórios em Dispositivos Móveis". In: XVI Congresso Brasileiro de Automática - CBA 2006, 2006, Salvador. Proceedings of the XVI CBA 2006, 2006. p. 482-487.

▪ PEROZZO, Reiner F.; PEREIRA, C. E. "Framework para Construção de Sistemas Supervisórios em Dispositivos Móveis". In: 24º Simpósio Brasileiro de Redes de Computadores - VIII Workshop de Tempo Real, 2006, Curitiba. Anais do WTR 2006, 2006. v. 1. p. 133-136.

▪ PEROZZO, Reiner F.; PEREIRA, C. E. "Arquitetura para Construção de Sistemas Supervisórios em Dispositivos Móveis". In: Iberian Latin American Congress on Computacional Methods in Engineering - XXVII CILAMCE, 2006, Belém. Proceedings of the XXVII CILAMCE, 2006.

REFERÊNCIAS

ANDRADE, S. S. **Sistemas Distribuídos de Supervisão e Controle Baseados em Componentes de Tempo-Real**. 2006. 215 p. Dissertação (mestrado) – Programa de Pós-Graduação em Mecatrônica, Universidade Federal da Bahia, Salvador, 2006.

APACHE. **Apache Software Foundation**. Disponível em: <http://www.apache.org>. Acesso em: 15 jan. 2007.

ARGO. **ArgoUML Project Home**. Disponível em: <http://argouml.tigris.org>. Acesso em: 28 nov. 2006.

AURRECOECHEA, C.; CAMPBELL, A. T.; HAUW, L. A Survey of QoS Architectures. **Multimedia Systems Journal**, Special Issue on QoS Architecture, ACM-Springer, p. 138 – 151, 1998.

AXEDA **Axeda Systems**, Inc. (ITA Group) Becomes Wizcon Systems. Disponível em: <http://www.automation.com/store/p1030details16085.php?x=1&pagePath=>. Acesso em 27 fev. 2007.

BASTO, V.; FREITAS, V. Distributed QoS Multimedia Transport. In: INTERNACIONAL CONFERENCE DFMA, 6-9 Feb. 2005, Besan. **Proceedings…** New York: IEEE Computer Society, p.15 – 21, 2005.

BECKER, L. B.; PEREIRA, C. E. SIMOO-RT: an Object-Oriented Framework for the Development of Real-Time Industrial Automation Systems. In: **IEEE Transactions On Robotics & Automation.** Los Angeles, v. 18, n. 4, p. 421 – 430, 2002.

BORCOCI, E.; et al. Service Management for end-to-end QoS Multimedia Content Delivery in Heterogeneous Environment. In: ADVANCED INDUSTRIAL CONFERENCE ON TELECOMMUNICATIONS, Lisboa**. Proceedings…** New York: IEEE Computer Society, p. 46 –52, 2005.

BORLAND. **Borland: Delphi**. Disponível em: <http://www.borland.com/br/products/delphi/index.html>. Acesso em: 15 jan. 2007.

DI NITTO, E.; SASSAROLI, G.; ZUCCALÀ, M. Adaptation of Web Contents and Services to Terminals Capabilities: the @Terminals approach. In: INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS, 23-26 Mar. 2003, Dallas-Fort Worth. **Proceedings…** New York: IEEE Computer Society, p. 433 – 440, 2003.

DONG-GI LEE; DEY, S. Dynamic Image Adaptation Technique and Architecture to Enhance Server Performance in Wireless Image Services. In: IEEE INTERNATIONAL SYMPOSIUM ON PERSONAL, INDOOR AND MOBILE RADIO COMMUNICATIONS, 5-8 Sept. 2004, Barcelona. **Proceedings…** New York: IEEE Computer Society, p. 3030 – 3035, 2004.

ELIPSE. **Elipse Software**. Disponível em: <http://www.elipse.com.br>. Acesso em: 2 dez. 2006.

ELUTIONS. **Wizcon News** Flash - Newsletter - Sept. 2006. Disponível em: <http://www.wizcon.com/news/newsletters/2006/ELUTIONS%20Newsletter%20-%20September%202006.htm>. Acesso em 27 fev. 2007.

FEI, YU; WONG, V.W.S.; LEUNG, V.C.M. Efficient QoS Provisioning for Adaptive Multimedia in Mobile Communication Networks by Reinforcement Learning. In: INTERNATIONAL CONFERENCE ON BROADBAND NETWORKS, 2004. San José. **Proceedings…** New York: IEEE Computer Society, p. 579 – 588, 2004.

FERRARI, D. The Tenet Experience and the Design of Protocols for Integrated Services Internetworks. **Multimedia Systems Journal**, Secaucus, v.6, n.3, p. 179 – 185, 1996.

GAMMA, E. et al. **Padrões de projeto:** soluções reutilizáveis de software orientado a objetos, Porto Alegre: Bookman, 2000. 364 p. ISBN: 85-7307-610-0.

GOPALAKRISHNA, G.; PARULKAR, G. Efficient Quality of Service in Multimedia Computer Operating Systems In: **Department of computer science**, Washington: Washington University. Report WUCS-TM-94-04, 1994.

GUEDES, G. T. A. **UML 2 Guia de Consulta Rápida**. São Paulo: Novatec, 2005. 109 p. ISBN: 85-7522-065-9.

HESSELMAN, C.; EERTINK, H.; PEDDEMORS, A. Multimedia QoS Adaptation for Inter-Tech Roaming. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, 3-July. 2001, Hammamet. **Proceedings…** New York: IEEE Computer Society, p. 554 – 560, 2001.

HOMESYSTEMS. **Homesystems:** ambientes inteligentes. Disponível em: <http://www.homesystems.com.br>. Acesso em: 22 jan. 2007.

IBM. **IBM Rational Software**. Disponível em: <http://www-306.ibm.com/software/rational>. Acesso em: 15 jan. 2007.

IBM J9. **IBM Software:** websphere everyplace micro environment. Disponível em: <http://www-306.ibm.com/software/wireless/weme>. Acesso em: 15 jan. 2007.

ICONICS. **Pocket GENESIS:** mobile pocket PC industrial software. Disponível em: <http://www.iconics.com/products/pocketgenesis.asp>. Acesso em: 03 dez. 2006.

ICONICS-UDS. **Universal Development System**. Disponível em: <http://www.mysoftwareautomation.com/news/press_release_display.asp?Release=ISA00_GENESIS-61.HTM>.Acesso em 27 fev. 2007.

JAVA SE. **Java SE Development Kit**. Disponível em: <http://java.sun.com/javase/downloads/index.jsp>. Acesso em: 15 jan. 2007.

KAPLAN, A.; LUNN, J. FlexXML: Engineering a More Flexible and Adaptable Web. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY: coding and

computing, Apr. 2001. Las Vegas. **Proceedings…** New York: IEEE Computer Society, p. 405 – 410, 2001.

MATSUI, Y. et al. An Extensible Object Model for QoS Specification in Adaptive QoS Systems. In: IEEE INTERNATIONAL SYMPOSIUM ON OBJECT-ORIENTED REAL-TIME DISTRIBUTED COMPUTING, 2-5 May, 1999. Saint Melo. **Proceedings...** New York: IEEE Computer Society, p. 129 – 132, 1999.

MICROSOFT. **Windows XP**. Disponível em: http://www.microsoft.com/brasil/windowsxp/default.mspx>. Acesso em: 15 jan. 2007.

MORAES, C. C.; CASTRUCCI, P. L. **Engenharia de Automação Industrial**. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2001. 295 p. ISBN: 85-216-1269-9.

MUCHOW, J. W. **J2ME Tecnologia & MIDP**. São Paulo: Makron Books, 2004. 581 p. ISBN: 85-346-1522-5.

MYSQL. **Open Source Database**. Disponível em: <http://www.mysql.com>. Acesso em: 15 jan. 2007.

NAHRSTEDT, K.; SMITH, J. Design, Implementation and Experiences of the OMEGA **End-Report** (MS-CIS-95-22), Pennsylvania: University of Pennsylvania, 1995.

OMG. **Object Management Group**. Disponível em: <http:// http://www.omg.org>. Acesso em: 28 nov. 2006.

PENDER, T. **UML a Bíblia.** Rio de Janeiro: Elsevier, 2004. 711 p. ISBN: 85-352-1408-9.

PEROZZO, R. F.; PEREIRA, C. E. Framework for Building Supervisory Systems in Mobile Devices. In: IEEE INTERNATIONAL CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION, 20-22 Sept. 2006. Praga. **Proceedings…** New York: IEEE-ETFA. p. 167-172, 2006.

ROGERS, G. F. **Framework-based Software Development in C++**, New Jersey: Prentice-Hall, 1997. 382 p. ISBN: 0-13-533365-2.

RUMBAUGH, J. et al. **Modelagem e Projetos Baseados em Objetos**. Rio de Janeiro: Campus, 1994. 652 p. ISBN:85-7001-841-X.

SHALLOWAY, A.; TROTT, J. R. **Explicando Padrões de Projeto:** uma nova perspectiva em projeto orientado a objeto. Porto Alegre: Bookman,2004. 328 p. ISBN: 85-363-0403-0.

SILVA, A.P.G.; SALVADOR, M. **O que são Sistemas Supervisórios?** RT 025.04. 12/2005. Disponível em: <http://www.elipse.com.br/download/download/artigos/rt025.04.pdf>. Acesso em: 28 jan. 2006.

SILVEIRA, P. R.; SANTOS, W. E. **Automação e Controle Discreto**. São Paulo: Érica, 1998. 229 p. ISBN: 85-7194-591-8.

SOFT BRASIL. **Utilização das Tecnologias Web Para Controle e Supervisão**. Disponível em: <http://www.softbrasil.com.br/artigos2.asp?id=18> Acesso em: 27 fev. 2007.

SUN WTK. **Sun Java Wireless Toolkit**.
Disponível em: <http://java.sun.com/products/sjwtoolkit>. Acesso em: 15 jan. 2007.

TIAN, M. et al. A Concept for QoS Integration in Web Services. In: INTERNATIONAL CONFERENCE ON WEB INFORMATION SYSTEMS ENGINEERING WORKSHOPS, 2003, Roma. **Proceedings...** New York: IEEE Computer Society, p. 149- 155, 2003.

TIBOLA, L. R. **Geração de Sistemas Supervisórios a partir de Modelos Orientados a Objetos.** 2000. 64 p. Dissertação (mestrado) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2000.

VP-UML. **Visual-Paradigm for UML**. Disponível em:
<http://www.visual-paradigm.com/product/vpuml>. Acesso em: 28 nov. 2006.

WIZCON. **The Complete Web-based Solution for Control and Information**. Disponível em: <http://www.wizcon.com>. Acesso em: 1 dez. 2006.

YAMAZAKI, T. A Study of QoS-based Multimedia Data Retrieval Systems. In: **IEEE International Symposium on Industrial Electronics**, 9-11 June, 2003, Rio de Janeiro. p. 485 – 489, 2003.