Ricardo Luis Balieiro

DESENVOLVIMENTO DE UMA FERRAMENTA COMPUTACIONAL PARA AQUISIÇÃO VIA INTERNET DE DADOS DE DISPOSITIVO DE CAMPO EM AMBIENTES FIELDBUS

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Engenharia Mecânica.

Área de Concentração: Dinâmica das máquinas e sistemas

ORIENTADOR: Prof. Dr. Luís Carlos Passarini

São Carlos
2008

AGRADECIMENTOS

Agradeço primeiramente a nosso Senhor Jesus Cristo e sua mãe Maria pela graça concedida de poder completar meu mestrado. Diante de todas as dificuldades, suas bênçãos trouxeram pessoas ou criaram situações que de alguma forma me apoiaram e auxiliaram nesses momentos.

À Smar Equipamentos Industriais Ltda. por todo apoio e incentivo à pós-graduação, em particular ao Libânio Carlos de Souza e Marco Aurélio de O. Pagnano, e a todos os amigos da Divisão de Desenvolvimento Eletrônico.

Ao meu orientador, Professor Dr. Luís Carlos Passarini, pela grande generosidade, dedicação e atenção dispensada em todos os momentos deste programa de mestrado, em especial na condução desta dissertação.

Sou imensamente grato a minha esposa Susana e aos meus filhos Gabriela e Felipe, pelo amor, compreensão e apoio, principalmente neste período em que muitas vezes estive ausente. Amo vocês.

RESUMO

BALIEIRO, R. L. (2008). Desenvolvimento de uma ferramenta computacional para aquisição via Internet de dados de dispositivo de campo em ambientes fieldbus. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2008.

A crescente utilização de equipamentos inteligentes na área de automação industrial tem assegurado eficiência e qualidade na produção. Em conseqüência deste crescimento, existe uma grande quantidade de equipamentos interligados abrangendo áreas geográficas distantes. Este cenário criou a demanda de sistemas que permitissem o acesso às informações geradas no chão de fábrica a partir de qualquer computador localizado na área industrial ou em qualquer outro ponto do planeta. A Internet propiciou uma nova opção para gerenciamento e monitoramento de equipamentos inteligentes a longa distância. Assim, este trabalho propõe um algoritmo para monitorar, via Internet, os eventos gerados por equipamentos inteligentes instalados em plantas industriais. Com o monitoramento contínuo, é possível analisar o desempenho dos equipamentos, detectar problemas e tomar decisões de forma a garantir que toda uma malha de controle não venha a parar inesperadamente.

Palavras chave: aquisição via Internet, Web services, dispositivos de campo, fieldbus, OPC.

ABSTRACT

BALIEIRO, R. L. (2008). Developing a data acquisition software tool via Internet for field devices in fieldbus networks. M.Sc. Dissertation – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2008.

The increasing usage of smart field devices in the industrial automation area has assured efficiency and quality of production. Thanks to this expansion, a large number of devices can be interconnected through distant geographical areas. This scenario created the need of computational systems that would make the information from the plant floor accessible to any computer in the local industrial area or any other earth wide location. The Internet has created a new possibility of remote managing and monitoring of smart devices. This work proposes an algorithm to monitor, via Internet, the events generated by smart devices installed in industrial plants. The continuous monitoring makes it possible to analyze the performance of the devices, detect problems and make decisions to ensure that an entire control loop does not come to an unexpected halt.

Keywords: data acquisition via Internet, Web services, field devices, fieldbus, OPC.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AI - Analog Input

AJAX - Asynchronous Javascript And XML

AO - Analog Output

API - Application Programming Interface

ASP - Active Server Pages

bps - bits per second

BUS - Business

CLP - Controlador Lógico Programável

CLR - Common Language Runtime

COM - Component Object Model

CORBA - Common Object Request Broker Architecture

CPU - Central Processing Unit (Unidade Central de Processamento)

DA - Data Access

DB - Database

DCOM - Distributed Component Object Model

DLL - Dynamic-Link Library

ERP - Enterprise Resource Planning

FF - Foundation™ Fieldbus

HDA - Historical Data Access

HSE - High-Speed Ethernet

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

IEC - International Electrotechnical Commission

IHM - Interface Homem Máquina

IIS - Internet Information Services

IP - Internet Protocol

ISP - Interoperable Systems Project

LAN - Local Area Network

LAS - Link Active Scheduler

MES - Manufacturing Execution Systems

MTS - Microsoft Transaction Server

NIC - Network Interface Card

OOSE - Object-Oriented Software Engineering

OPC - OLE for Process Control

PID - Proportional Integral Differential

PLC - Programmable Logic Controller

RMI - Remote Method Invocation

RPC - Remote Procedure Call

SCADA - Supervisory Control And Data Acquisition System

SOA - Service-Oriented Architecture

SOAP - Simple Object Access Protocol

SPG - Setpoint Ramp Generator

SSL - Secure Socket Layer

TCP - Transmission Control Protocol

UML - Unified Modeling Language

UDDI - Universal Description, Discovery and Integration

XML - eXtensible Markup Language

XMLDA - XML Data Access

W3C - World Wide Web Consortium

WorldFIP - World Factory Instrumentation Protocol

WSDL - Web Services Description Language

SUMÁRIO

1 INTRODUÇÃO

Desde a Revolução Industrial introduzida na indústria têxtil da Inglaterra em 1750, o trabalho braçal vem sendo substituído pelo trabalho das máquinas; em um primeiro momento por máquinas a vapor, depois por motores de explosão e em seguida por motores de energia elétrica. Para Stenerson (2002), as máquinas não só substituem os músculos do homem como também seu sistema sensorial e sua capacidade de ação, sendo capaz de executar operações em sistemas complexos de produção ou de serviços em tempo real. Essa evolução das máquinas originou novos desenvolvimentos industriais que fazem parte de uma nova revolução industrial, denominada "Automação Industrial". A Automação Industrial visa, principalmente, a produtividade, a qualidade e a segurança em um processo.

Para Souza e Oliveira (2006), o início da automação aconteceu na década de 20 nas linhas de montagem automobilística de Henry Ford e, desde então, a automação industrial, impulsionada pelos avanços tecnológicos, tem crescido cada vez mais. Na década de 60 surgiram os primeiros Controladores Lógicos Programáveis ou PLCs. Com o avanço da microeletrônica e o surgimento nos anos 90 de programas de computadores, foi possível obter uma maior produtividade, qualidade, competitividade e uma maior integração entre o chão de fábrica e o ambiente corporativo. Esse cenário impulsionou os fabricantes de PLCs a produzirem não apenas Sistemas Supervisórios de Controle e Aquisição de Dados (Sistemas SCADA), como também aplicativos mais

especializados, proporcionando maior desempenho, modularidade e expansibilidade nas soluções de automação.

Este ritmo de transformação atingiu também o setor de manutenção industrial onde houve a substituição de equipamentos puramente mecânicos por equipamentos com base eletromecânica e dispositivos microprocessados (dispositivos de campo inteligentes). Estes dispositivos de campo inteligentes passaram a ter uma Unidade Central de Processamento (CPU) embutida, de modo a não só fornecer ou atuar no processo como também armazenar históricos, conter algoritmos de controle e se auto-diagnosticar.

Estes dispositivos passaram a ser interligados em nível de chão de fábrica por redes de comunicação digital, denominadas "barramento de campo" ou *Fieldbus*, e vêm substituindo o sinal analógico 4-20mA existentes ainda hoje na indústria. A tecnologia digital permitiu que os dispositivos realizassem intercâmbio de informações necessárias à coordenação do funcionamento da planta; possibilitou a interconectividade entre produtos de diferentes fabricantes; e permitiu que um sistema coletasse um maior número de informações da planta industrial para que tivesse um ganho de produtividade no sistema como um todo. Esta facilidade na capacidade de comunicação gerou o aparecimento de vários padrões de protocolos de comunicação, tais como *Foundation™ Fieldbus*, *Profibus*, *Can Open*, etc, que na maioria das vezes são incompatíveis entre si.

Os fabricantes de equipamentos passaram a desenvolver *drivers* para a troca de informações entre os dispositivos de campo. Isto fez com que somente através do *driver* do fabricante fosse possível acessar os dados existentes em um equipamento causando grandes dificuldades para a comunicação entre diferentes fornecedores e equipamentos. Para contornar esse problema de comunicação ocorrida pela falta de padronização entre fabricantes, foi criada em 1996 a *OPC Foundation*, que estabelece regras e padrões de

comunicação entre aplicações. Este protocolo é hoje o padrão de fato da indústria (IWANITS, 2002).

Um servidor OPC fornece dados em tempo real provenientes de sensores, executa comandos de controle, gera status de comunicação, dados de desempenho, estatística do sistema, etc. O OPC foi projetado originalmente baseado na tecnologia Microsoft® COM/DCOM, permitindo assim a comunicação com um simples computador ou entre aplicações em múltiplos computadores em uma mesma rede. Em conjunto com o DCOM, o OPC possibilitou a integração de aplicações e o compartilhamento de dados sobre a Rede de Área Local ou LAN, criando assim um canal de informação rico entre o chão de fábrica e os sistemas de controle e supervisórios ou SCADA.

Existe hoje na indústria uma outra preocupação que consiste em não somente monitorar e controlar, mas também diagnosticar problemas antes que eles aconteçam na planta industrial. Nos últimos anos, tem-se investido em sistemas de monitoramento contínuo de dispositivos de campo de plantas industriais de modo a garantir que toda uma malha de controle não venha a parar inesperadamente (PAGNANO, 2003).

Esta necessidade nasceu a partir do crescimento acelerado da utilização de equipamentos inteligentes na automação industrial. Estes equipamentos muitas vezes estão operando em barramentos de campo que podem se estender por grandes áreas geográficas da área industrial (WARNER; KAGHAZCHI, 2007).

Gerenciar ativos garante uma maior disponibilidade dos equipamentos, reduz a variabilidade do processo e os custos com manutenção, ou seja, obtém-se maior tempo e qualidade de produção e, conseqüentemente, reduz-se perdas.

Com uma solução para gestão de ativos é possível analisar o desempenho dos equipamentos, prognosticar problemas, delimitar soluções alternativas e ações para a otimização, além de melhorar o planejamento estratégico (PAGNANO, 2003).

Se o sistema diagnosticar que um dispositivo de campo está com problemas ou prestes a parar, pode-se efetuar sua substituição de forma a minimizar o tempo de parada de uma planta industrial.

Diante deste cenário fica evidente a necessidade de criar sistemas industriais integrados, cujo fluxo de informações passe do chão de fábrica aos sistemas corporativos de forma eficaz e sem problemas de integração, além de permitir que estas informações sejam acessadas por qualquer computador localizado na área industrial ou de qualquer outro lugar do planeta. Este tipo de integração está cada vez mais presente na área de automação depois que o sistema Windows™ foi adotado pela indústria como o sistema operacional padrão para automação industrial.

Segundo Marino (2000), pelo menos 70% dos computadores existentes hoje no chão de fábrica utilizam Windows NT como sistema operacional, permitindo utilizar uma arquitetura totalmente integrada com o protocolo TCP/IP como principal meio de transporte de informação. Então, utilizar tecnologias de supervisão e controle via redes TCP/IP, ou seja, a mesma tecnologia hoje empregada na Internet, torna mais fácil a integração do chão de fábrica com os diversos sistemas existentes, fazendo com que o fluxo de informações seja eficiente, diminuindo os custos de integração e manutenção.

A partir dos anos 90, com a expansão acelerada da Internet e a possibilidade de enviar grandes quantidades de dados de um lado a outro do mundo, a computação distribuída passou a receber mais atenção dos desenvolvedores de software. A Internet pode ser vista como uma opção de transferência dos dados para monitoramento remoto.

Para casos de controle em tempo real[1], a Internet apresenta empecilhos por não se poder prever quanto tempo levará para executar um comando, porém para sistemas onde o tempo de resposta não é um fator crítico, como sistemas de gerenciamento de ativos, a Internet mostra-se um caminho promissor. Todo este contexto gerou a necessidade de conectar não somente clientes finais com as informações geradas no chão de fábrica através de um navegador Web, como também conectar estas informações com outros sistemas utilizando a Internet. Sistemas de conectividade têm exigido cada vez mais interoperabilidade entre plataformas.

Diante dessa nova necessidade surgiram os *Web Services*[2] como tecnologia de integração que pode funcionar através de várias plataformas. Os *Web Services* estão alicerçados em padrões da indústria como o XML[3] e o SOAP[4], e representam a próxima etapa lógica na evolução das tecnologias distribuídas.

1.1 JUSTIFICATIVA

Segundo Warner e Kaghazchi (2007), o crescimento acelerado da utilização de equipamentos inteligentes na automação industrial tem assegurado a eficiência e qualidade de produção. Estes equipamentos muitas vezes estão operando em barramentos de campo em paralelo em um mesmo local, ou em outro local no chão de fábrica. Como o chão de fábrica pode se estender por grandes áreas geográficas, surge a necessidade de assegurar um modo fácil e útil de monitorar os dados fornecidos pelos equipamentos.

---

[1] Tempo real é a habilidade de um sistema em executar um comando ou instrução e disponibilizar a resposta em um intervalo de tempo bem definido.
[2] *Web Services* são componentes que permitem às aplicações enviar e receber dados em formato XML.
[3] XML é uma especificação técnica, desenvolvida pela W3C, para escrever documentos de forma estruturada.
[4] SOAP é um protocolo para troca de informações estruturadas utilizando tecnologias baseadas em XML.

Além do monitoramento de grandes áreas geográficas, outra necessidade crescente é a de permitir que as informações geradas na área industrial possam ser acessadas por outros sistemas ou pessoas de fora do ambiente industrial.

Graças a sua expansão acelerada, aliada às novas tecnologias surgidas nos últimos anos, a Internet se apresenta como opção de monitoramento e integração de sistemas. Entre as novas tecnologias estão os *Web Services*, considerados como um passo à frente na computação distribuída e cada vez mais utilizado em âmbito corporativo.

Efetuando o monitoramento contínuo de dispositivos de campo é possível analisar o desempenho dos equipamentos, prognosticar problemas, delimitar soluções alternativas e ações para garantir que toda uma malha de controle não venha a parar inesperadamente.

Neste sentido, surge como motivação deste trabalho o desenvolvimento de uma aplicação que permita o monitoramento e armazenamento de dados de dispositivos de campo via Internet. Este monitoramento e armazenamento podem ser efetuados de qualquer lugar da área industrial ou fora dela, já que utiliza a Internet como meio de comunicação. Com os dados obtidos será possível realizar uma análise para prognosticar problemas que possam ocorrer, e corrigi-los antes que toda uma malha industrial venha a parar, acarretando grande perda de produção.

1.2 OBJETIVO GERAL DO TRABALHO

O objetivo deste trabalho é o desenvolvimento de um software que possibilite o monitoramento de dados, através da Internet, de equipamentos de campo instalados em plantas industriais.

1.3 OBJETIVO ESPECÍFICO

O objetivo específico é testar e validar o funcionamento do protótipo, e também analisar seu desempenho, contrapondo os resultados obtidos com os requisitos propostos para o protótipo. Para avaliar o desempenho, o protótipo será submetido a duas situações: o “caso crítico” de desempenho (utilizando linha discada) e o “melhor caso” (utilizando rede Ethernet).

1.4 DESCRIÇÃO SUCINTA DA METODOLOGIA EMPREGADA

O protótipo será composto por dois módulos: módulo cliente e módulo servidor. Para a formação de cada módulo, propõe-se o desenvolvimento de vários componentes. Esta estratégia de dividir os módulos em componentes possibilita que os componentes sirvam como plataforma de comunicação para outros softwares. O trabalho aborda as características de cada componente, o contexto envolvido no agrupamento dos componentes para a formação dos módulos, e os aspectos envolvidos na comunicação entre os módulos.

1.5 ORGANIZAÇÃO DO TRABALHO

Esta dissertação está estruturada da seguinte forma: o Capítulo 2 apresenta os principais benefícios e características do barramento de campo Foundation™ Fieldbus; a evolução dos protocolos de comunicação e o surgimento da tecnologia OPC; os principais aspectos envolvidos nas aplicações distribuídas e a arquitetura dos serviços de componente COM+. Ainda no Capítulo 2, são discutidos os tópicos relevantes aos

Web Services, sua utilização na aplicação distribuída e seus protocolos de troca de mensagens entre aplicações; e por fim, os aspectos de utilização e mecanismos de comunicação de várias estratégias propostas para a aquisição de dados de plantas industriais através da Internet.

O Capítulo 3 mostra um estudo comparativo entre as principais abordagens utilizadas na obtenção de dados via Internet e propõe uma estrutura para a aquisição de dados, que é o objetivo deste trabalho. Em seguida, os requisitos para a criação do protótipo são analisados, e os modelos UML utilizados para o desenvolvimento do protótipo e aspectos envolvidos na codificação do aplicativo são descritos. As principais infra-estruturas e configurações utilizadas para o teste do protótipo também são apresentadas neste capítulo.

O Capítulo 4 discute a série de testes realizada para identificar o desempenho do protótipo e os resultados obtidos, e descreve também um estudo de aplicação onde se propõe a utilização concreta do protótipo em um problema real de uma determinada empresa.

Por fim, o Capítulo 5 apresenta as conclusões sobre o trabalho desenvolvido e sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

2.1 A ARQUITETURA FOUNDATION™ FIELDBUS

A *Fieldbus Fondation*[5] é uma organização independente, sem fins lucrativos, composta de mais de cem dos principais fornecedores e usuários de controle e instrumentação do mundo. A *Fieldbus Foundation* foi criada em 1994 pela união de dois consórcios internacionais, o *Interoperable Systems Project* (ISP) e o *World Factory Instrumentation Protocol* (WorldFIP), com o objetivo de criar um padrão internacional compatível com as normas da *International Electrotechnical Commission* (IEC) que pudesse ser utilizado amplamente pela indústria de automação e controle de processos (MATA, 2005). Foi então criada a tecnologia *Foundation™ Fieldbus*[6].

O *Foundation™ Fieldbus* é uma rede local (LAN) para automação industrial totalmente digital e bidirecional que conecta equipamentos *Fieldbus* e instrumentação de controle de processos (ARQUITETURA..., 2008). O protocolo *Foundation™ Fieldbus* (FF) é um padrão aberto que permite processamento distribuído, diagnóstico avançado e redundância.

Diferente de outras tecnologias, como HART ou Profibus onde um sistema de controle centralizado executa as operações de controle e supervisão, o FF delega esta

[5] Fieldbus Fondation é uma organização dedicada à evolução da tecnologia Foundation™ Fieldbus.

função para um dos equipamentos de campo distribuído na rede (PERES FILHO; MATA, 2004). A rede FF permite conectar 32 equipamentos de campo, cada um com um endereço único na rede, sendo 12 equipamentos alimentados pelo próprio barramento e os 20 equipamentos restantes não alimentados pelo barramento (Figura 01). Para evitar que haja um tráfego alto na rede, o ideal é não ultrapassar o total de 16 equipamentos conectados à rede (MATA, 2005).

Há dois tipos primitivos de equipamentos definidos:

- *Link Master devices (LM)*;
- *Basic devices*.

Em um dado instante, a rede FF possui apenas um *Link Active Scheduler* (LAS) que controla o tráfego de dados cíclicos e acíclicos na rede. Caso um LAS falhe, um equipamento do tipo *Link Master* se torna um LAS, assegurando assim o funcionamento da rede (Figura 01). Equipamentos do tipo *Basic*, por serem mais simples, não podem ser promovidos a LAS (PERES FILHO; MATA, 2004).

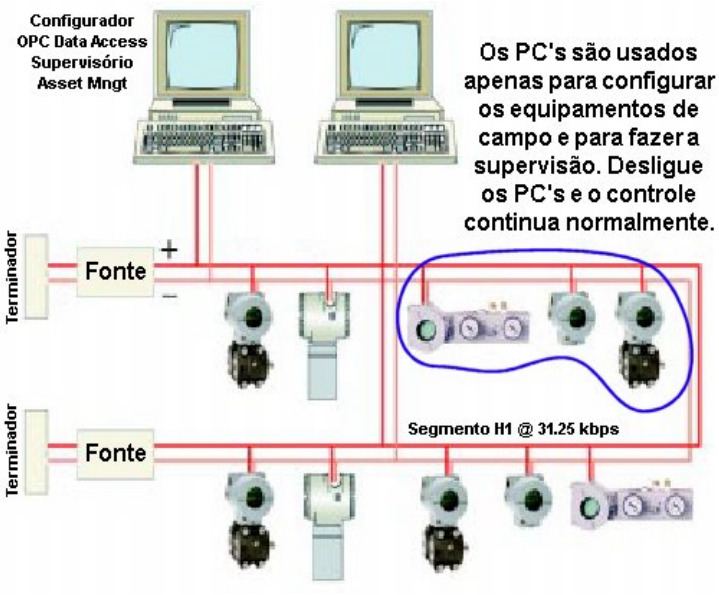

Figura 01 - Topologia típica de instalação FF
Fonte: Peres Filho e Mata (2004)

[6] Foundation™ Fieldbus é um sistema de comunicação totalmente digital.

O meio físico de uma rede FF é composto de um par de fios trançados e blindados por onde passa tanto a alimentação como a comunicação de dados, podendo atingir a distância total em torno de 1,8 km na rede H1[7]. As normas IEC61158 e ISAS50.02 são utilizadas para definir o meio físico de uma rede FF. A modulação do sinal é feita pelo transmissor através da variação de uma corrente de 10 mA a 31,25 kbit/s, em uma carga equivalente de 50 ohms, resultando em uma tensão modulada de 1 V sobreposta à tensão contínua de alimentação (DC) do barramento. Cada equipamento FF consome tipicamente 20 mA (Figura 02).

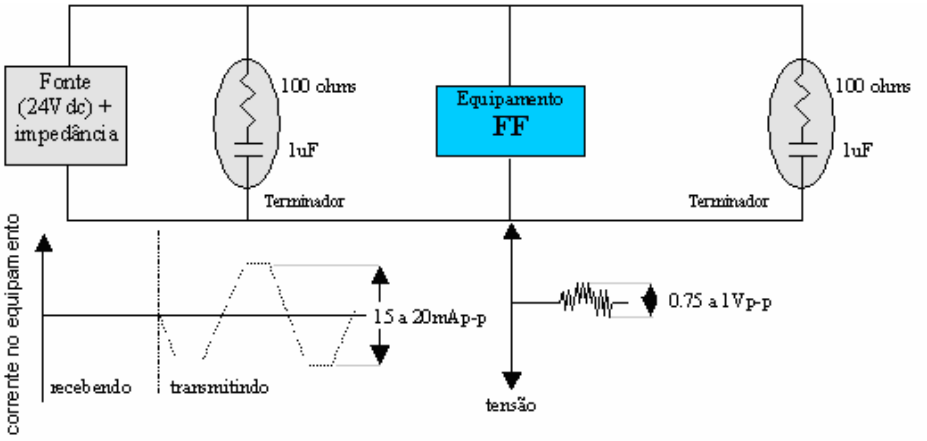

Figura 02 - Simplicidade da camada física FF H1 (IEC 61158)
Fonte: Mata (2005)

Nas primeiras especificações da norma *Foundation™ Fieldbus*, havia a norma H2 além da norma H1. Devido à evolução da tecnologia, a H2 foi substituída pelo HSE, que usa Ethernet a 100 Mbps baseada na norma ETHERNET (IEEE802.3-2000, ISO/IEC8802.3-2000). Desta forma, é possível construir uma rede de controle industrial utilizando Ethernet na rede FF HSE com equipamentos de diversos fabricantes comunicando-se a taxas de 100 Mbps, podendo chegar a 1 Gbps ou até mesmo ao novo padrão 10 Gbps (Figura 03).

[7] Foundation™ Fieldbus H1 é uma das versões do protocolo Foundation™ Fieldbus.

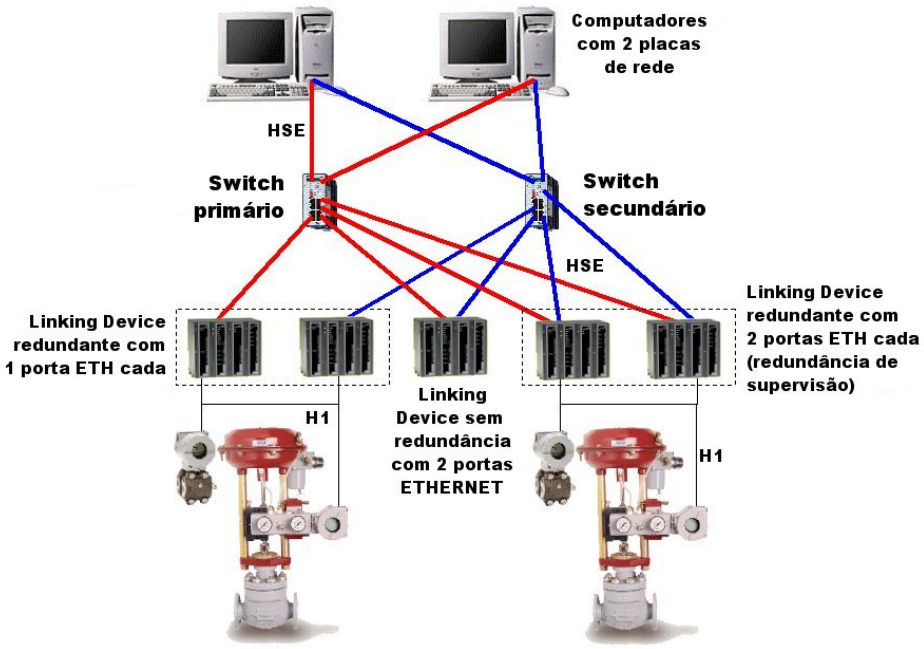

Figura 03 - Topologia de rede FF HSE redundante tipo estrela
Fonte: Mata (2005)

O *Foundation™ Fieldbus* utiliza blocos de funções pré-definidos para descrever as funções de um dispositivo e definir um acesso uniforme aos dados, e por isso essa tecnologia é normalmente chamada de tecnologia orientada a blocos, ou *Block Oriented Technology*. Os blocos funcionais contidos dentro de um dispositivo permitem descrever as informações sobre as tarefas que o dispositivo deve executar e suas capacidades e características, como por exemplo, um controlador PID, um totalizador ou uma entrada analógica. A FF define o comportamento e as interfaces dos blocos funcionais de forma a ter uma Linguagem Universal para a descrição de aplicações de controle de processos e automação (MATA, 2005).

Os blocos são compostos por algoritmos e parâmetros de entrada (*input*), por onde os dados chegam ao algoritmo, e parâmetros de saída (*output*), que são os resultados do processamento ou de controle (*contained*) (Figura 04).

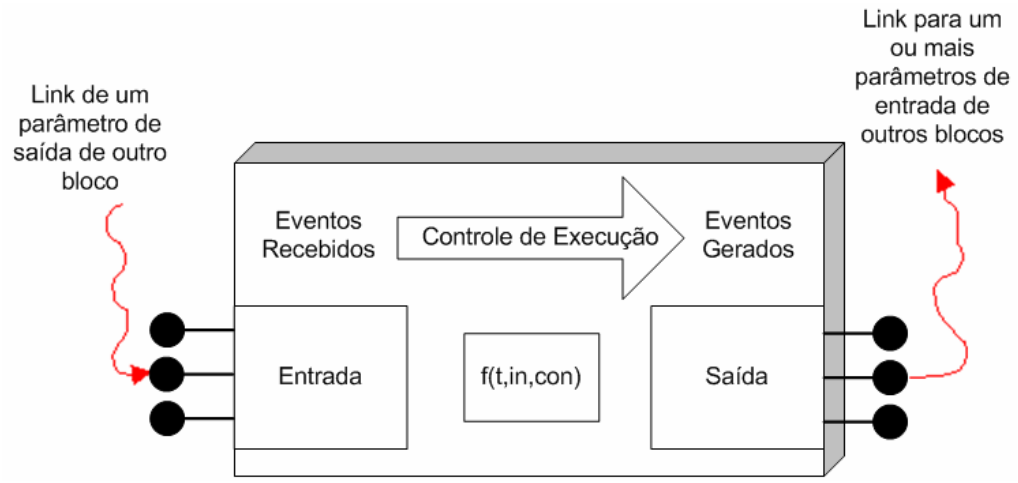

Figura 04 - Estrutura de um bloco funcional Foundation™ Fieldbus
Fonte: Mata (2005)

Isso permitiu criar uma biblioteca de blocos padrão FF com as aplicações básicas de controle de processos e automação, mas a norma FF permite que sejam criados novos blocos de funções específicas (lógica *fuzzy*, redes neurais, *gateways*) e que sejam adicionadas funcionalidades ou parâmetros específicos a blocos padrões. Esses são chamados de blocos melhorados (*enhanced*) ou avançados (*advanced*).

Os blocos FF são classificados em três tipos básicos (MATA, 2005):

- **Blocos de recursos (*Resource Block*):** cada equipamento possui apenas um bloco de recurso. O bloco de recurso é padronizado e descreve as características físicas do dispositivo, tais como ID do fornecedor, versão do dispositivo, características e capacidade de memória.
- **Blocos funcionais (*Function Blocks*):** os blocos funcionais descrevem as funcionalidades do dispositivo e definem como elas podem ser acessadas. Cada bloco tem um objetivo além de suas próprias entradas e saídas. Cada dispositivo é equipado com pelo menos um bloco funcional.

- **Blocos transdutores (*Transducer Blocks*):** os blocos transdutores conectam blocos funcionais ao mundo externo, isolando-os das características específicas de cada hardware. Eles podem ser usados para calibrar, deslocar medidas, posicionar dados, linearizar características e converter unidades físicas.

Para Peres Filho e Mata (2004), o protocolo *Fieldbus Foundation* proveu benefícios à indústria de automação moderna que podem ser resumidos em:

- **Interoperabilidade:** permite que um equipamento de um fabricante seja substituído por outro na mesma rede FF mantendo as características do original. Isto permite aos usuários mesclar equipamentos de campo e sistemas de vários fornecedores.
- **Flexibilidade:** blocos funcionais são uma linguagem universal de descrição de aplicações que pode ser portada para diferentes plataformas.
- **Controle distribuído:** devido à tecnologia de ponta existente nos dispositivos, as tarefas são divididas para tornar o sistema mais eficiente e mais simples.
- **Diagnóstico avançado:** devido à inteligência distribuída em cada instrumento, é possível monitorar e registrar informações quanto à integridade dos equipamentos. O pessoal da planta pode executar a manutenção pró-ativa sem esperar uma parada programada, evitando ou reduzindo assim o tempo ocioso da planta.

2.2 O PADRÃO DE COMUNICAÇÃO OPC

2.2.1 VISÃO GERAL DO OPC

Segundo Kondor (2008), no passado, cada fabricante de software desenvolvia sua própria interface ou *driver* para trocar dados com dispositivos de hardware no campo utilizando protocolos de comunicação, proprietários ou não (Figura 05). Para Shimanuki (1999), este tipo de arquitetura resulta em vários problemas:

- Duplica esforços devido à necessidade de se construir mais de um módulo de software para um único dispositivo;
- Conflito de acesso, porque um dispositivo não pode ser acessado simultaneamente por mais de uma aplicação;
- O módulo de software precisa ser atualizado toda vez que ocorrer uma atualização do dispositivo.

Para resolver estes problemas, em 1995, as empresas Fisher-Rosemount, Intellution, Intuitive Technology, Opto22, Rockwell e Siemens Ag uniram-se em um grupo de trabalho chamado *OPC Task Force*. A Microsoft providenciou o apoio técnico para a nova tecnologia (IWANITS; LANGE, 2002). O objetivo da *OPC Task Force* foi desenvolver um padrão de acesso a dados em tempo real, utilizando o sistema operacional Windows, baseado na tecnologia OLE/DCOM.

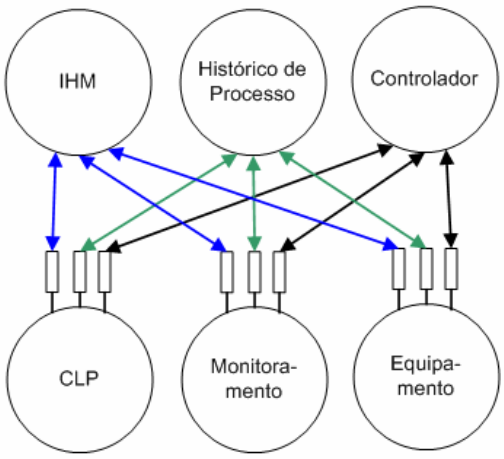

Figura 05 - Soluções proprietárias
Fonte: Kondor (2008)

Um ano depois, em agosto de 1996, foi disponibilizada a primeira especificação da norma, chamada *OPC Specification Version 1.0*. Após a liberação da versão 1.0 do OPC, foi fundada a *OPC Foundation*[8].

Utilizando a tecnologia OPC, cada fabricante de hardware disponibiliza somente um *driver* Servidor OPC para seus equipamentos, definindo uma interface de alto desempenho. A tecnologia OPC, segundo Iwanits e Lange (2002), traz como benefícios:

- A eliminação de *drivers* de comunicação proprietários com a padronização das interfaces de comunicação entre os servidores e clientes de dados em tempo real, facilitando a integração e manutenção dos sistemas (Figura 06);
- Melhora o desempenho e otimiza a comunicação entre os diversos dispositivos de automação industrial;
- Possibilita a interoperabilidade entre sistemas de gestão empresarial (ERP), de execução de manufatura (MES) e aplicações Windows (Excel, etc.);

[8] A OPC Foundation é uma instituição dedicada a atender as necessidades da indústria através do aprimoramento e ampliação da especificação OPC, garantindo assim a interoperabilidade de sistemas de automação.

- Reduz custos e tempo de desenvolvimento e, como conseqüência, reduz o custo de integração de sistemas.

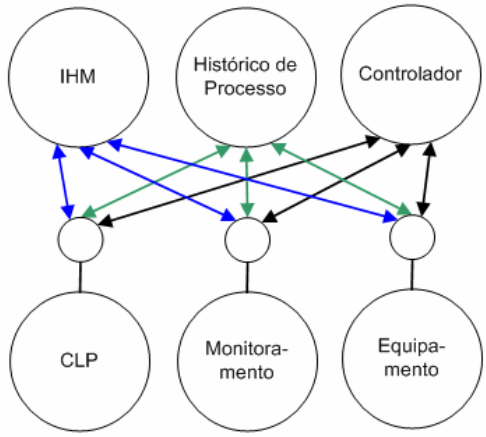

Figura 06 - Solução OPC
Fonte: Kondor (2008)

2.2.2 TECNOLOGIA OPC

O protocolo OPC é baseado no modelo de componentização criado pela Microsoft e denominado *Distributed Component Object Model* (DCOM), de modo que todas as definições de DCOM, tais como objetos, interfaces, métodos, etc, são aplicadas ao OPC. Atualmente existem as seguintes especificações publicadas ou em processo de aprovação, indicadas na Tabela 01 (IWANITS; LANGE, 2002).

Tabela 01 - Especificação OPC

Especificação	Conteúdo	Versão
OPC Overview	Descrição geral dos campos de aplicação das especificações OPC.	***Versão 1.00***
OPC Common Definitions and Interfaces	Definição das funcionalidades básicas para as demais especificações.	***Versão 1.00***
OPC Data Access Specification	Definição da interface para leitura e escrita de dados em tempo real.	***Versão 2.05***
OPC Alarms and Events Specification	Definição da interface para monitoração de eventos.	***Versão 1.02***
OPC Historical Data Access Specification	Definição da interface para acesso a dados históricos.	***Versão 1.01***

Especificação	Conteúdo	Versão
OPC Batch Specification	Definição da interface para acesso aos dados de processos por batelada (*batch*). Esta especificação é uma extensão da *OPC Data Access Specification*.	***Versão 2.00***
OPC Security Specification	Definição da interface para utilização de políticas de segurança.	***Versão 1.00***
OPC and XML	Integração entre OPC e XML para aplicações via Internet (Web).	***Versão candidata 1.05***

Fonte: Iwanits e Lange (2002)

A Figura 07 mostra como estão compostas as diversas interfaces existentes no OPC e os relacionamentos entre elas. Estas interfaces têm como objetivo atender às aplicações nos mais variados campos. Elas podem ser utilizadas independentes umas das outras e também podem ser combinadas em uma aplicação (IWANITS; LANGE, 2002).

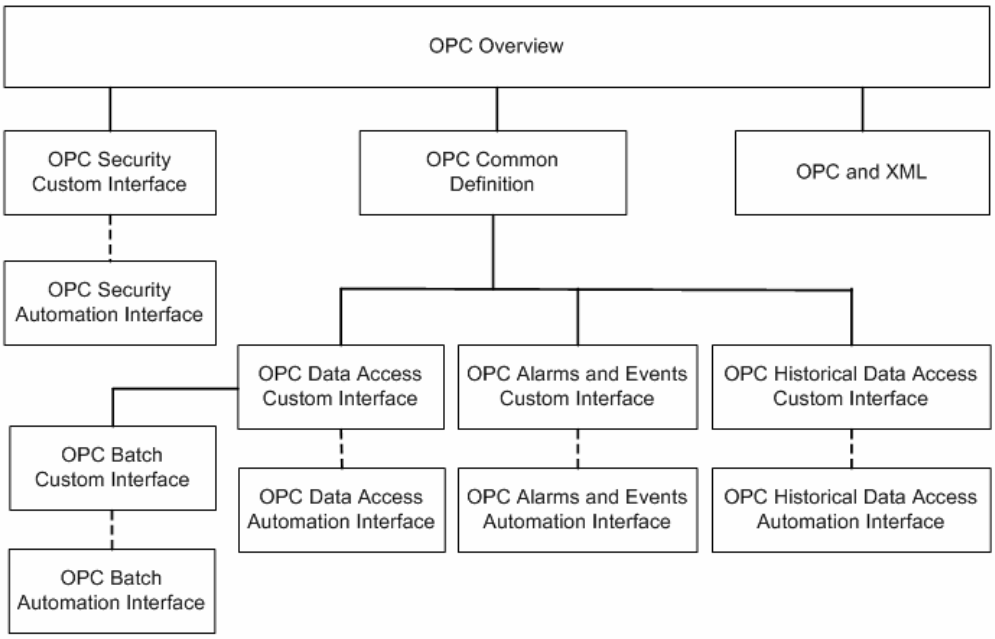

Figura 07 - Interfaces OPC
Fonte: Iwanits e Lange (2002)

As aplicações que utilizam o protocolo OPC são classificadas de acordo com sua funcionalidade: *Cliente OPC* ou *Servidor OPC*. Os servidores OPC são aplicações que respondem a requisições de um ou mais clientes OPC através dos mecanismos padrão

do OPC para ler, escrever e atualizar dados diretamente nos dispositivos de campo. Os clientes OPC normalmente são produtos de controle e monitoramento de dados.

2.2.3 ASPECTOS DE UTILIZAÇÃO DO PADRÃO OPC

As aplicações clientes OPC podem ser escritas nas mais diversas linguagens de programação, como *C++*, *Visual Basic*, *DOT.NET*, linguagens de *script*, etc. A aplicação cliente pode acessar o servidor OPC através de duas interfaces: *OPC Custom Interface* e *OPC Automation Interface* (Figura 08). O acesso ao servidor através da interface *Custom* pode ser feito por aplicações desenvolvidas em linguagens que suportam as chamadas de funções por ponteiros, tais como: *C*, *C++*, *Delphi*, etc. Outras linguagens como *Visual Basic*, *DOT.NET* e *VBA* não têm suporte a ponteiros de funções. Nestes casos, é utilizada a interface *OPC Automation*, que funciona como um *proxy*[9] entre as aplicações clientes e a interface *Custom* (IWANITS; LANGE, 2002).

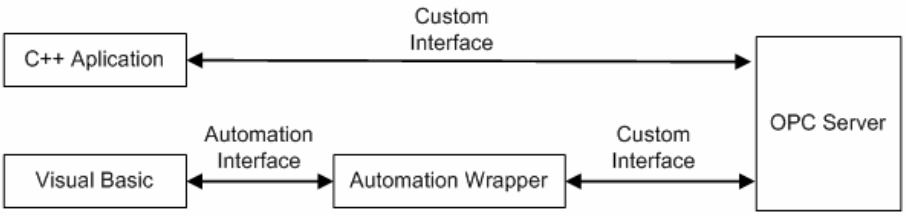

Figura 08 - Acesso aos servidores OPC usando interfaces *Custom* e *Automation*
Fonte: Iwanits e Lange (2002)

A maioria dos fabricantes tem adotado o OPC Data Access, ou apenas OPC DA, como padrão de transferência de dados do processo entre hardware e software, para assegurar a escalabilidade[10] em aplicações de automação industrial. Para obter os dados do processo, o OPC DA oferece os seguintes objetos: *Servidor*, *Grupo* e *Item*.

[9] Um proxy funciona como intermediário entre duas entidades.
[10] Escalabilidade é a característica de um sistema de manipular uma porção crescente de trabalho de forma uniforme, ou estar preparado para o crescimento do mesmo.

Segundo Rehbein (2008), para obter dados do processo, o primeiro objeto a ser criado é o *Servidor OPC*. Tipicamente, o OPC DA contém um Servidor OPC, e este Servidor OPC pode servir vários Clientes OPC. A responsabilidade do Servidor OPC é gerenciar a aquisição dos dados requisitados do dispositivo do controle de processo pelo cliente do OPC. O objeto Servidor OPC cria os *Grupos OPC*, que são uma coleção de dados a serem obtidos (Figura 09). O cliente pode utilizar os Grupos OPC para organizar seu acesso aos dados e, por exemplo, criar um Grupo OPC para agrupar todos os valores de um processo de pressão e um outro para os valores de temperatura.

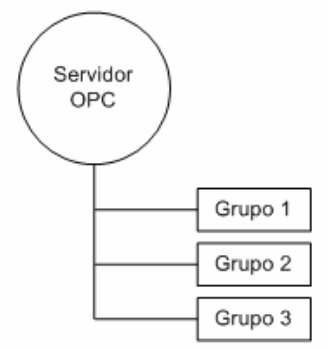

Figura 09 - Relacionamento entre *Servidor* e *Grupos*
Fonte: Rehbein (2008)

Objetos *Item OPC* são adicionados dentro dos Grupos OPC pela aplicação Cliente OPC. É através dos objetos Item OPC que os dados são disponibilizados. Os objetos Item OPC não podem ser acessados diretamente, mas somente através de objetos Grupo OPC (Figura 10).

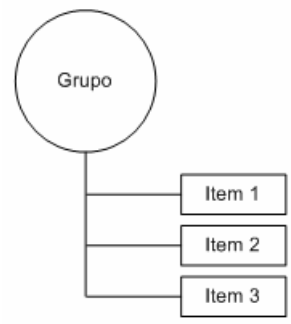

Figura 10 - Relacionamento entre *Grupos* e *Itens*
Fonte: Rehbein (2008)

2.2.4 MECANISMO DE COMUNICAÇÃO

Segundo Iwanits e Lange (2002), o OPC fornece quatro tipos de mecanismo de acesso ao OPC Data Access: *synchronous*, *asynchronous*, *subscription* e *refresh*.

- ***Synchronous*:** Quando o cliente chama o método de leitura de dados, o *Servidor OPC* trava a aplicação cliente até que os dados requisitados pelo objeto *Grupo* sejam lidos por uma única instrução e retornados para o cliente. Este tipo de leitura pode bloquear a aplicação cliente por um longo período.
- ***Asynchronous*:** Quando o cliente chama o método de leitura de dados, o *Servidor OPC* não trava a aplicação cliente. Os dados do objeto *Grupo* são lidos e retornados para o cliente através de um mecanismo de *call-back*.
- ***Subscription*:** Este método de leitura envolve a geração do evento *Data Change* pelo *Servidor OPC*. Este evento é gerado de tempos em tempos, quando mudanças significativas ocorrem nos dados configurados no objeto *Grupo*.
- ***Refresh:*** Este método de leitura envolve a geração do evento *Data Change* pelo *Servidor OPC*. Esta função força a geração do evento *Data Change* a qualquer momento.

Os dados retornados pelo Servidor OPC possuem três propriedades: *Value*, *Quality* e *TimeStamp*:

- ***Value:*** representa o último valor do item lido no instrumento e armazenado pelo *Servidor OPC* no *cache*[11].

[11] Cache é um dispositivo de armazenamento de dados.

- ***Quality:*** representa quão confiável é o valor do *Item* em *cache* no *Servidor OPC*. Os valores para *Quality* podem ser: *Good*, *Bad* ou *Uncertain*.
- ***TimeStamp:*** representa o horário do mais recente acesso do *Servidor OPC* ao dispositivo de campo para obter os valores dos itens requisitados.

2.3 HISTÓRICO DAS APLICAÇÕES DISTRIBUÍDAS

2.3.1 COMPONENTES DE UMA APLICAÇÃO

Segundo Building... (2002) as aplicações de computadores possuem três áreas de funcionalidades, independente de como foram implementas ou do que fazem. Estas áreas são classificadas como:

- Interface com o usuário ou outro sistema: permite à aplicação comunicar-se com o meio externo;
- Lógica de negócio ou regras de negócio: parte do processo de computação que descreve as operações, definições e restrições para alcançar os objetivos desejados;
- Acesso a dados: código que automatiza o armazenamento, procura e recuperação de dados pelas aplicações.

2.3.2 APLICAÇÕES MONOLÍTICAS

Para a Building... (2002), durante muitos anos as aplicações eram monolíticas, tanto para *mainframes* quanto para computadores pessoais. Uma aplicação monolítica implementa a interface com o usuário, a lógica do negócio e o acesso a dados em um único bloco. Este modelo de aplicação tinha várias desvantagens, entre elas:

- A necessidade de computadores grandes e caros quanto maior fosse a aplicação;
- Grandes aplicações monolíticas tendem a ter maior complexidade, maior dificuldade de desenvolvimento e maior custo de manutenção;
- Pouca facilidade de integração com outros aplicativos, mesmo que estes aplicativos tivessem sido escritos na mesma linguagem e estivessem rodando no mesmo computador.

2.3.3 APLICAÇÕES CLIENTE-SERVIDOR

Muitos destes problemas foram resolvidos com o surgimento de aplicações cliente-servidor. Neste modelo, a aplicação é dividida entre dois computadores. As interfaces com o usuário e as regras de negócio ficam no computador *Cliente*, e o acesso a dados no computador *Servidor*. Kirtland (1998) diz que apesar deste modelo ter diminuído drasticamente os problemas em relação às aplicações monolíticas, não resolveu o problema de reusabilidade[12].

[12] Reusabilidade é o potencial que um componente tem de ser reusado.

2.3.4 APLICAÇÕES DISTRIBUÍDAS

O modelo de componentes surgiu com a intenção de ajudar a resolver os problemas encontrados com os modelos de aplicações monolíticas e cliente-servidor, e também permitir que se criassem componentes de software reutilizáveis. A modelagem orientada a objetos (COM) permite que duas ou mais aplicações, ou componentes, cooperem facilmente entre si, independente de terem sido escritos na mesma linguagem ou de serem de fornecedores diferentes (KIRTLAND, 1998).

O lançamento do DCOM em 1995 permitiu que componentes fossem distribuídos em máquinas diferentes e pudessem ser invocados remotamente. Para aumentar os recursos do DCOM, em 1998, a Microsoft lançou o Servidor de Transações da Microsoft ou MTS. O MTS disponibilizou novos serviços de componentes, tais como gerenciamento de transação, segurança declarativa baseada em funções, entre outros (LÖWY, 2001).

Com a chegada do Windows 2000, a Microsoft incorporou um novo conjunto de serviços de componentes denominado *COM+ 1.0*, que evoluiu para o *COM+1.5* com o Windows XP. Este novo conjunto oferece instrumentação, gerenciamento de variáveis compartilhadas, segurança, comunicação entre processos, *pooling* de componentes, facilidade no gerenciamento e distribuição de pacotes, entre outras funcionalidades (LÖWY, 2001).

2.3.5 ARQUITETURA DOS SERVIÇOS DE COMPONENTE COM+

Para Löwy (2001), COM+ são serviços fornecidos pelo sistema operacional para prover um ambiente gerenciado para processos. Este ambiente proporciona às

aplicações corporativas maior robustez, escalabilidade e reutilização de componentes. Os serviços fornecidos pelo COM+ são utilizados em tempo de execução[13] por componentes em aplicações distribuídas, tais como aplicações Cliente/Servidor. Estes componentes ou aplicações COM+ são gerenciados pelo *Component Service Explorer* (Figura 11).

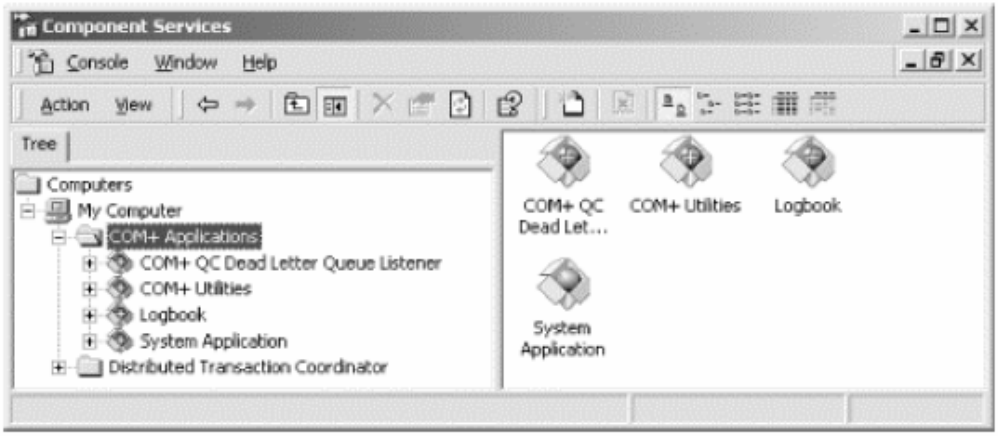

Figura 11 - *Component Service Explorer*
Fonte: Löwy (2001)

De acordo com a Building... (2002), para utilizar os serviços fornecidos pelo COM+ é necessário criar um componente, ou seja, uma DLL, e configurá-lo com os serviços que se deseja utilizar. Em seguida, este componente deve ser instalado dentro do *Component Service Explorer*. A instalação é feita criando-se uma *Aplicação COM+* no *Component Service Explorer* e adicionando o componente (DLL) dentro dessa *Aplicação COM+*. Uma *Aplicação COM+* pode conter uma ou mais classes de um ou mais componentes (DLL), conforme ilustrado na Figura 12.

[13] Tempo de execução é o período em que um programa de computador permanece em execução.

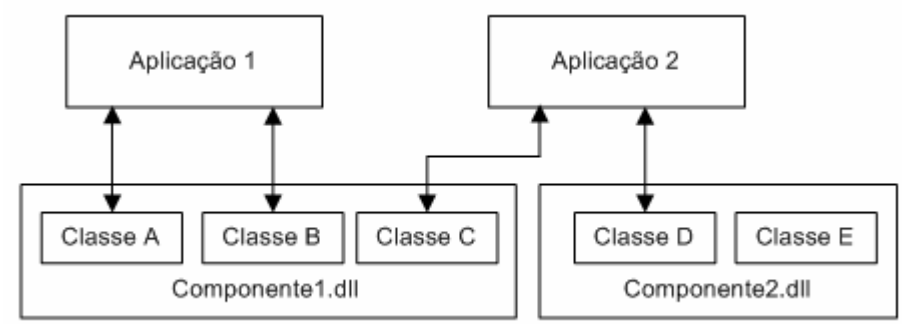

Figura 12 - Estrutura de aplicação COM+
Fonte: Building... (2002)

Os componentes ou DLLs não podem ser executados sozinhos, eles só podem ser executados de duas formas: no modo *Biblioteca* ou modo *Servidor*. No modo *Biblioteca*, a DLL é executada no mesmo processo do cliente que a criou. No modo *Servidor*, a DLL é executada dentro de um processo hospedeiro chamado *dllhost.exe* (Figura 13).

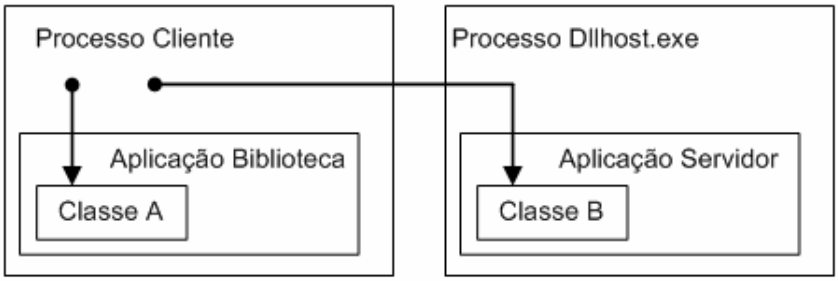

Figura 13 - Modo de execução de componente COM+
Fonte: Building... (2002)

Building... (2002) diz que os serviços fornecidos pelo COM+ são implementados em termos de contexto. O contexto é um ambiente de execução de um processo que fornece serviços em tempo de execução para um ou mais objetos resididos ali. Quando um objeto dentro de um contexto chama um objeto em outro contexto, o método é interceptado por um *proxy*. O *proxy* permite que o COM+, em tempo de execução, pré-processe a invocação e execute os serviços necessários solicitados pelo código que chamou o objeto, como mostrado na Figura 14.

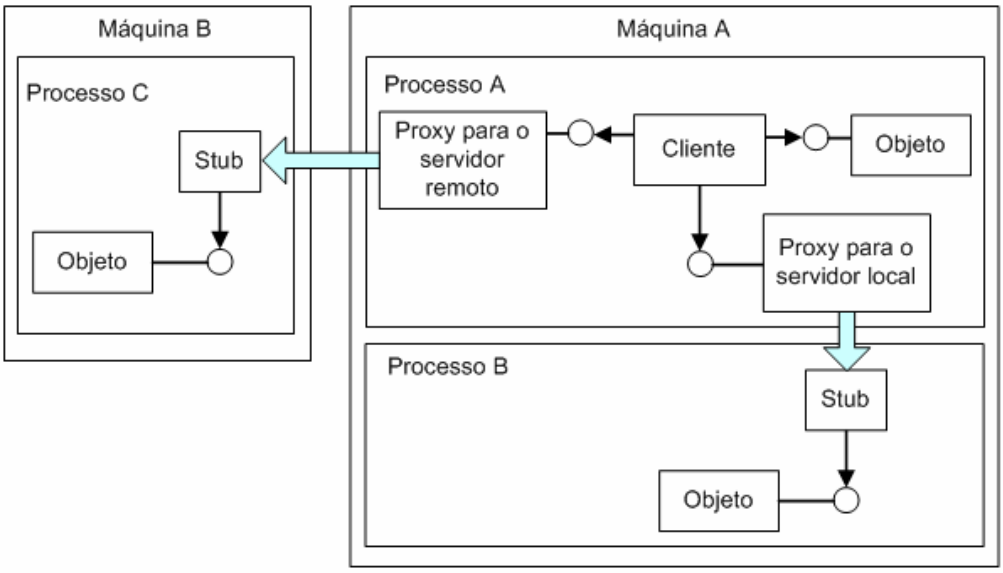

Figura 14 - Contexto e interceptação
Fonte: Löwy (2001)

O contexto e a interceptação podem se expandir de acordo com as necessidades da aplicação. Conforme novos serviços são solicitados pelos componentes, o COM+ expande o contexto para comportar os novos serviços solicitados (MICROSOFT, 2002).

Para usar os serviços em tempo de execução do COM+, a classe deve ser marcada com atributos declarativos. Cada serviço define um ou mais atributos declarativos que são usados para controlar seu comportamento. O COM+ armazena os valores dos atributos declarativos de uma classe em um repositório chamado *COM+ Catalog*. Quando um componente é ativado em tempo de execução, o COM+ checa o catálogo e verifica quais os requisitos necessários para ativar o componente e cria o contexto para comportá-los. Os dados armazenados no catálogo podem ser alterados ou atualizados usando o *Component Service Explorer* ou através de APIs administrativas.

2.3.6 COM+ E A PLATAFORMA MICROSOFT .NET FRAMEWORK

Em julho de 2000, a Microsoft anunciou oficialmente sua nova plataforma de desenvolvimento, o Microsoft .NET. A principal estrutura do .NET é a *Common Language Runtime* (CLR) que substitui a tecnologia COM mas não a tecnologia COM+. Para utilizar os recursos do COM+, o .NET faz uso de um *namespace* (bibliotecas de códigos do Microsoft.Net) chamada *System.EnterpriseServices* que contém todas as classes[14] requeridas para prover serviços do COM+ para uma aplicação. A Tabela 02 descreve os principais atributos encontrados no *namespace Enterprise Services*.

Tabela 02 - Principais atributos encontrados no *namespace Enterprise Services*

Serviços	Descrição
Synchronization	Controla automaticamente a concorrência requerida para um componente.
Transaction	Ao atualizar dois ou mais bancos de dados, este serviço provê gerenciamento protegido de transações[15] para as atualizações.
Just-in-Time (JIT) activation	Cria o objeto apenas quando um método é chamado e destrói o objeto quando o método for completado.
Object pooling	Permite criar previamente um conjunto de objetos para serem usados posteriormente.
Security	Permite ao desenvolvedor controlar o acesso aos seus componentes de acordo com a identificação do cliente.
Queued components	Permite que toda chamada de método de objeto seja gravada em uma fila dentro do sistema operacional. Esta fila é executada em algum momento no futuro.
Loosely Coupled Events	Permite que um objeto (chamado *Editor*) lance eventos, e um ou mais objetos (chamados *Assinantes*) recebam estes eventos.

Fonte: Building... (2002)

[14] Classe representa um conjunto de objetos com características afins.

2.4 WEB SERVICES

A importância e a aplicabilidade da computação distribuída têm aumentado significativamente nos últimos anos. O fator que mais contribuiu para esse aumento foi a Internet, pela sua simplicidade e conglomerado de redes em escala mundial de milhões de computadores interligados. Para Eide et al. (2001), a evolução dos modelos de programação acelerou o crescimento da Internet. Um desses modelos é o *Web Services*.

Web Services são considerados uma extensão do modelo de componentes para a Internet e, dessa forma, ideal para a computação distribuída baseada em componentes. Ainda segundo Eide et al. (2001), com a computação distribuída, em que a lógica dos aplicativos é particionada em unidades fisicamente distribuídas entre dois ou mais computadores, pode-se vincular diferentes organizações e unidades organizacionais. A lógica da aplicação pode ser reutilizada em várias aplicações. Parte de uma aplicação distribuída pode ser atualizada sem precisar parar a aplicação toda.

Eide et al. (2001) confirmam que CORBA e DCOM são os protocolos dominantes na computação distribuída permitindo a chamada de objetos remotos, mas tanto CORBA quanto DCOM foram projetados para aplicativos empresariais com software sendo executado na mesma plataforma e na mesma rede local. Quando o aplicativo é projetado para a Internet, estes dois protocolos são de grande complexidade de implementação, mostram problemas de escalabilidade, além de consumirem recursos valiosos do servidor. Neste ponto, os *Web Services* apresentam-se como a evolução das tecnologias distribuídas, baseadas em componentes.

Os *Web Services* tem como características:

[15] Transação é um conjunto de procedimentos que é executado em um banco de dados; o usuário vê uma transação como uma única ação.

- Fraco acoplamento[16] ao cliente;
- São do tipo *Stateless*, ou seja, cada chamada resulta em um novo objeto criado para o serviço que o requisitou;
- Tem como fundamento o SOAP.

Estas características fazem dos *Web Services* a tecnologia que permite uma maior escalabilidade e menor complexidade em termos de aplicação para a Internet.

2.4.1 CARACTERÍSTICAS DO WEB SERVICES

A proposta da tecnologia *Web Services* é a comunicação entre aplicações através de serviços oferecidos por componentes de um sistema distribuído. Estes componentes podem ser acessados dinamicamente através de uma rede. Os *Web Services* retornam dados em formato XML, diferente das páginas Web que retornam dados no formato HTML. Isto permite que os dados sejam descritos e manipulados com grande facilidade tanto por quem envia quanto por quem recebe.

Para disponibilizar serviços, os *Web Services* estão estruturados em uma série de padrões abertos e tecnologias amplamente difundidas. Estas tecnologias permitem expor regras de negócios que podem ser acessadas e entendidas por qualquer linguagem de programação, em qualquer sistema operacional, rodando em qualquer dispositivo. Esses padrões são:

[16] O acoplamento fraco mede quanto uma classe depende de ou está relacionada a outra classe ou subsistema.

- ***XML (*eXtensible Markup Language*):*** fornece a sintaxe comum para a representação de dados. O XML é uma especificação técnica desenvolvida pela W3C *(World Wide Web Consortium* - entidade responsável pela definição da área gráfica da Internet), e é definida como o formato universal para dados estruturados na Web. Esses dados consistem de tabelas, desenhos, parâmetros de configuração, etc. A linguagem então trata de definir regras que permitam escrever documentos que sejam interpretados adequadamente pelo computador;
- ***SOAP (Simple Object Access Protocol):*** fornece a semântica para a troca de dados. O protocolo SOAP é um protocolo para troca de informações estruturadas utilizando tecnologias baseadas em XML. Sua especificação define um formato que provê maneiras para se construir mensagens que possam trafegar através de diversos protocolos;
- ***WSDL (Web Services Description Language):*** fornece um mecanismo para descrever as capacidades de um *Web Service*. O WSDL é uma linguagem baseada em XML utilizada não somente para descrever os serviços dos *Web Services*, mas também para especificar como acessá-los e quais as operações ou métodos estão disponíveis;
- ***UDDI (Universal Description, Discovery and Integration):*** é uma especificação da indústria para publicar e localizar informações sobre *Web Services*. O UDDI é um serviço de registros onde provedores de serviços publicam seus *Web Services*. Os usuários de serviços utilizam o UDDI para encontrar serviços que lhes interessem e obter os metadados[17] necessários para utilizar esses serviços.

[17] Metadados representam dados sobre dados.

2.4.2 SOAP E WEB SERVICES

Segundo Cunha (2002), SOAP é o padrão para troca de mensagens entre aplicações e os *Web Services*, pois a construção desta tecnologia baseia-se em XML e HTTP. Para ter independência de plataforma e linguagem de programação, o protocolo SOAP foi estruturado para encapsular e transportar chamadas remotas de procedimento ou RPC. RPCs são chamadas locais a métodos de objetos (ou serviços) remotos. Portanto, é possível acessar serviços de um objeto localizado em um outro ponto da rede, através de uma chamada local a este objeto. Cada chamada ou requisição exige uma resposta.

Segundo Reckziegel (2006), as vantagens do SOAP são:

- É um padrão da indústria, criado por um consórcio do qual a Microsoft faz parte e adotado pela W3C e por várias outras empresas;
- Os dados são estruturados usando XML, portanto as mensagens podem ser compreendidas por quase todas as plataformas de hardware, sistemas operacionais e linguagens de programação;
- Pode ser usado em combinação com vários protocolos de transporte de dados, entre eles, o HTTP;
- Atravessa firewalls e roteadores;
- Pode ser usado tanto de forma anônima como com autenticação (nome/senha).

As desvantagens do SOAP são (RECKZIEGEL, 2006):

- Falta de interoperabilidade entre ferramentas de desenvolvimento do SOAP;
- Brechas de segurança. O SOAP não define um mecanismo de criptografia do conteúdo de suas mensagens;
- Não há garantia de que a mensagem vá chegar ao seu destino;
- O fato de utilizar HTTP e passar por firewalls também pode ser visto como um problema de segurança, já que aplicações não autorizadas poderiam acessar seus dados tanto internamente como externamente à rede.

A Figura 15 mostra a estrutura de uma mensagem do protocolo SOAP:

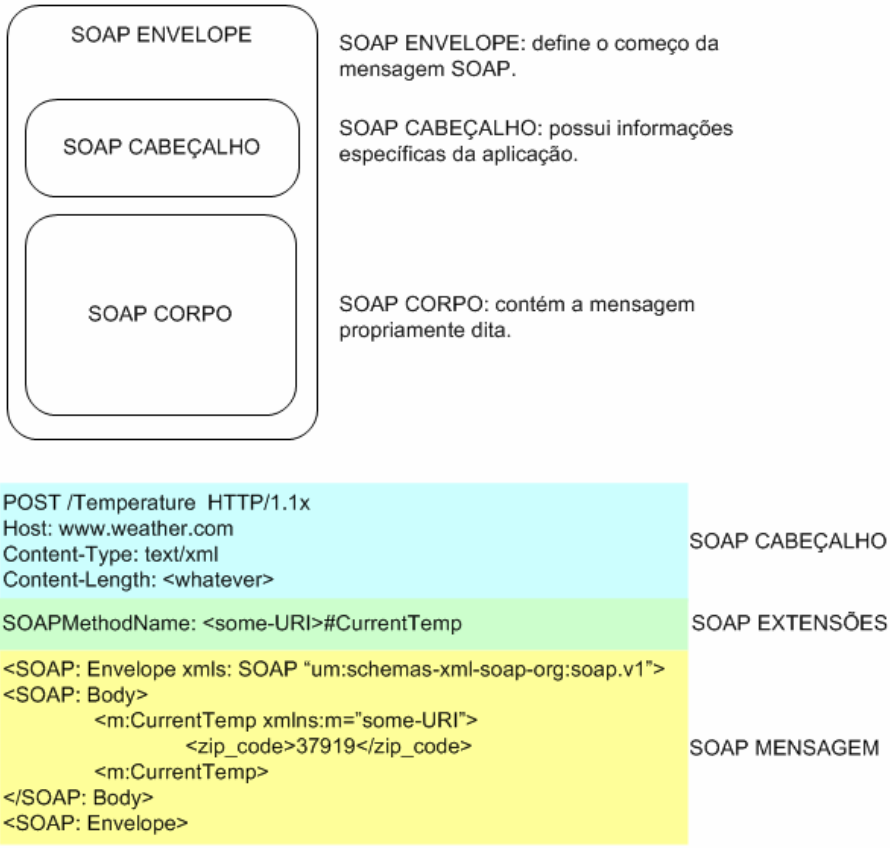

Figura 15 - Protocolo SOAP
Fonte: Eide et al. (2001)

2.5 ESTRUTURAS UTILIZADAS PARA TROCA DE DADOS ENTRE APLICATIVOS

Um fato importante observado nos dias de hoje é a necessidade da indústria de criar sistemas integrados que permitam que o fluxo de informações do chão de fábrica passe aos sistemas corporativos, e também que estas informações sejam acessadas por qualquer computador em qualquer local do planeta.

Como já há algum tempo que a Internet passou a ser utilizada dentro das redes internas das empresas e, mais recentemente, na automação industrial, nesta seção serão analisados projetos que têm em comum a troca de dados entre aplicativos e plantas industriais através da Internet. Os projetos são analisados e comparados em uma ordem que sugere a evolução da estrutura de um projeto para outro.

2.5.1 ESTRUTURA UTILIZANDO HTML - AJAX

Para Chau e Khai (2007), a possibilidade de acessar o chão de fábrica via Internet permite que além de receber, monitorar e exibir os valores de parâmetros fornecidos pelos barramentos de campo, seja possível transmitir sinais de controle para os dispositivos instalados na fábrica. Este conjunto de operações é característico de sistemas SCADA. Ainda segundo Chau e Khai (2007), estes dados podem ser acessados de duas formas: indireta e diretamente.

Na forma indireta, cria-se uma camada de código para extrair as informações do SCADA e disponibilizá-las na Internet (Figura 16). Este tipo de abordagem apresenta a desvantagem de necessitar de um especialista com grande conhecimento em SCADA para poder integrá-lo à Internet.

Na forma direta, cria-se um acesso direto ao *OPC - Data Access* (OPC DA) através de páginas Web, conforme também mostrado na Figura 16.

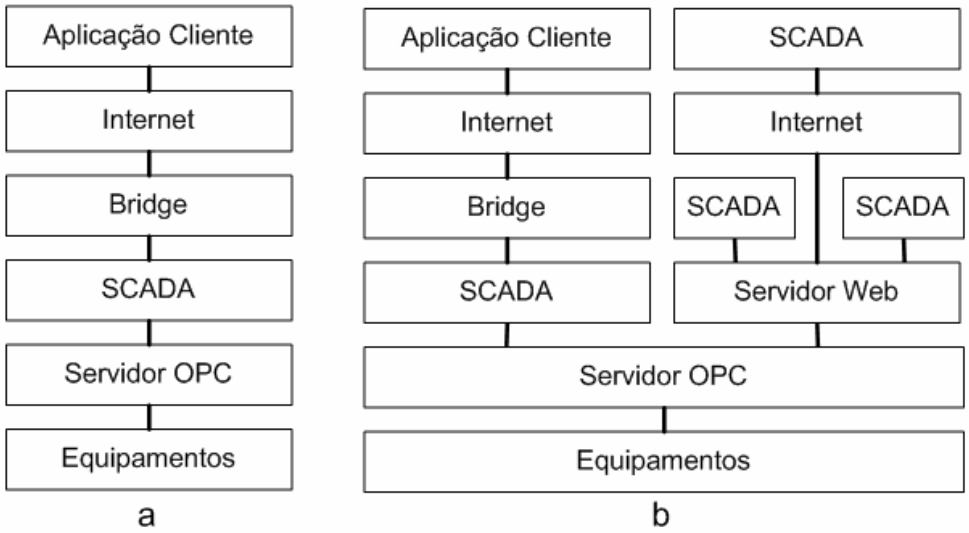

Figura 16 - Acesso Direto (a). Acesso Indireto (b)
FONTE: Chau e Khai (2007)

Focando no acesso direto, Chau e Khai (2007) propuseram duas possibilidades de implementação. A primeira seria criar uma página ASP com uma Biblioteca de Ligação Dinâmica, ou *ActiveX*[18] *DLL*[19]. Esta ActiveX DLL faria as chamadas ao Servidor OPC e retornaria o resultado para a página ASP, que por sua vez transformaria o resultado em Linguagem de Marcação de Hipertexto (ou HTML) e retornaria para o navegador Web do cliente (Figura 17).

Este tipo de abordagem apresenta um sério problema de atualização dos dados na tela do cliente. A atualização é feita automaticamente pela página HTML, que efetua de tempos em tempos uma nova requisição dos dados no IIS e todo o procedimento acima descrito é refeito. Este tipo de implementação apresenta baixo desempenho, desperdício de tempo gasto na atualização dos gráficos de tela, e carga excessiva do servidor e de banda com as buscas redundantes por novas informações no servidor Web.

[18] ActiveX são pequenos programas que podem ser incluídos dentro de páginas Web.
[19] DLL são arquivos com extensão DLL que podem conter códigos, dados e recursos (ícones, fontes, cursores, entre outros) em qualquer combinação.

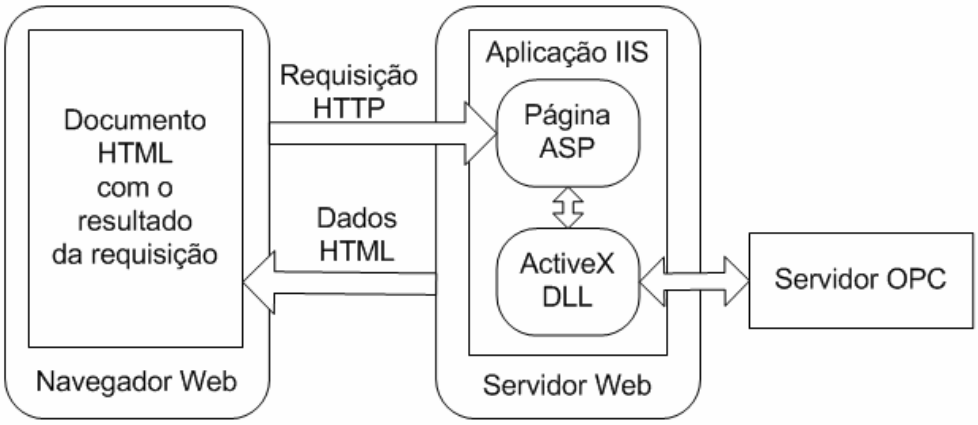

Figura 17 - SCADA Web baseado no IIS
FONTE: Chau e Khai (2007)

A segunda proposta adotada por Chau e Khai (2007) foi a criação de um sistema utilizando ASP.NET e AJAX. O AJAX possibilita efetuar solicitações assíncronas de informações no servidor e atualizar, na tela do cliente, apenas os dados que foram recebidos. Este procedimento evita a necessidade de atualização de toda a tela, resultando numa grande melhora no desempenho da aplicação e maior velocidade de atualização dos dados (Figura 18).

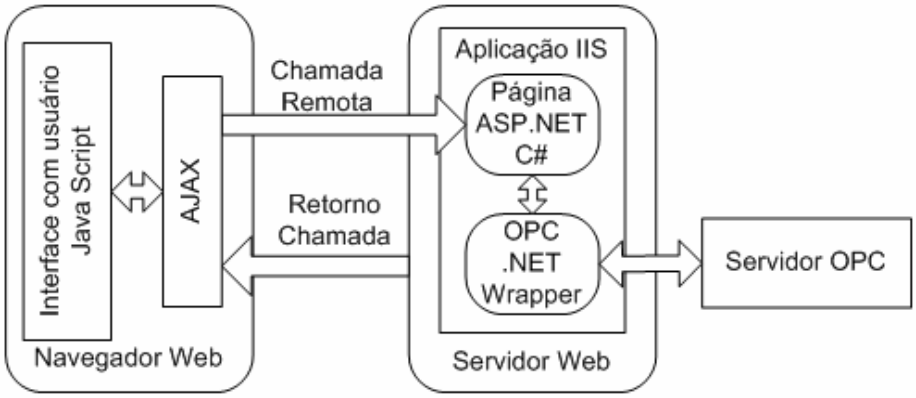

Figura 18 - SCADA Web utilizando ASP.NET e AJAX
FONTE: Chau e Khai (2007)

2.5.2 ESTRUTURA UTILIZANDO AJAX

Warner e Kaghazchi (2007) desenvolveram uma ferramenta para obter dados remotos, via Ethernet e Internet, de equipamentos que provêm dados de diagnósticos. O projeto utiliza Servidor OPC e interface *PROFIBUS*, e os dados obtidos são exibidos em um navegador Web. A Figura 19 exibe o diagrama de interação entre as camadas de software envolvidas na ferramenta de diagnóstico.

Cada camada converte o formato dos dados para o nível acima de forma que cada vez mais os dados fiquem em um formato genérico. Isto facilita a adição ou alteração de um tipo de barramento de campo sem causar impacto na exibição final.

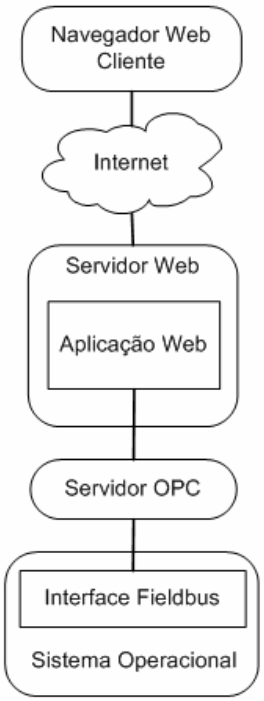

Figura 19 - Estrutura proposto por Warner e Kaghazchi
Fonte: Warner e Kaghazchi (2007)

A estrutura do software de Warner e Kaghazchi (2007) está exemplificada na Figura 20. Nos níveis inferiores, os *drivers* e o sistema operacional tratam o controle de hardware e a comunicação de rede. Nos níveis superiores, o servidor OPC, o servidor

Web e as páginas Web tratam a formatação e a apresentação dos dados no navegador Web do cliente, que pode ser o *Microsoft® Internet Explorer* ou *Mozila Firefox*™.

Foi utilizado um servidor OPC para *PROFIBUS* com funções para retornar dados de diagnóstico dos equipamentos conectados ao barramento de campo. Para a visualização dos dados, foi utilizada uma aplicação Web implementada para Microsoft ASP.NET com C#. As páginas baseiam-se na tecnologia AJAX, para reduzir a quantidade de dados transferidos para a atualização de tela, e Warner e Kaghazchi (2007) configuraram a taxa de atualização da página Web no AJAX em 10 segundos. A segurança da aplicação foi provida por encriptação SSL nos dados das páginas Web e *login* de usuário.

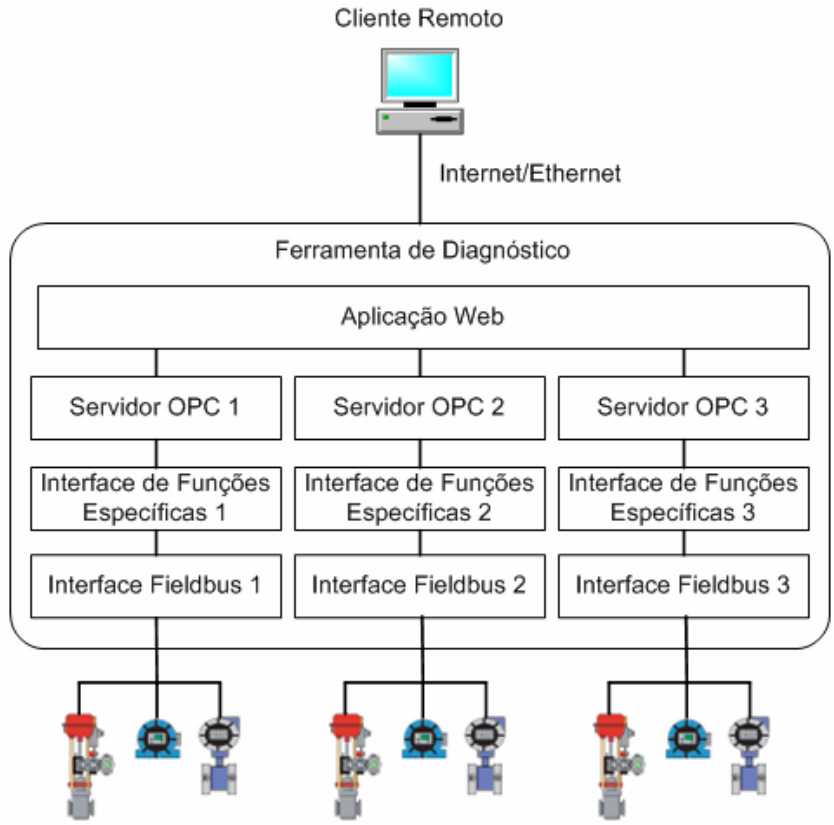

Figura 20 - Estrutura de software de Warner e Kaghazchi
Fonte: Warner e Kaghazchi (2007)

2.5.3 ESTRUTURA UTILIZANDO OPC XML-DA

Para Tan et al. (2007), as tecnologias Web têm grande importância para os sistemas de controle e monitoramento de plantas industriais. Aliado as tecnologias Web, tem se adotado o XML como padrão aberto para transferência de dados em várias plataformas baseadas no ambiente da Internet. Diante disso, Tan et al. (2007) propuseram um sistema distribuído de aquisição e monitoramento de dados baseados nas tecnologias OPC e XML.

Geralmente, uma arquitetura em três camadas é utilizada em integrações com a Web, como mostra a Figura 21. A camada inferior interliga os equipamentos de campo com o controlador, provendo dados e eventos para o sistema de automação. A camada do meio contém as regras de negócio e funções de comunicação entre os clientes da camada superior e o sistema de automação da camada inferior. Finalmente, a camada superior baseia-se no modelo Cliente-Servidor, usando o servidor Web como fonte de dados e o navegador Web como cliente.

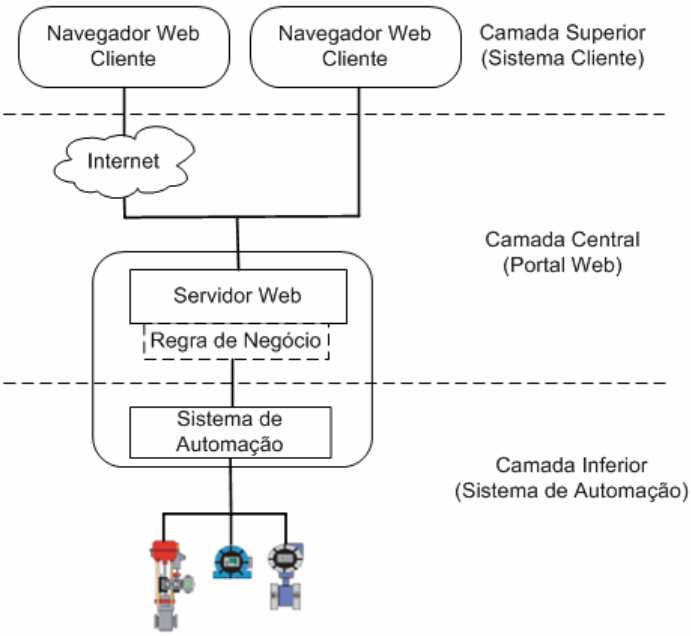

Figura 21 - Arquitetura genérica de integração Web
Fonte: Tan et al. (2007)

Segundo Tan et al. (2007), a camada inferior apresenta certa complexidade no momento da integração devido à grande quantidade de servidores OPC DA existentes no mercado. Cada servidor OPC DA pode apresentar uma semântica própria para acessar os dados além da grande quantidade de tipos de dados, dificultando o mapeamento entre os dados e a aplicação Web (Figura 22).

A especificação do OPC DA é um conjunto padronizado de objetos, métodos e propriedades COM que permitem a interoperabilidade para a automação do chão de fábrica, o controle de processos, as aplicações de monitoramento, etc. O OPC DA utiliza o DCOM para interligar o chão de fábrica a aplicações empresariais via rede Ethernet, mas, por causa disso, as aplicações OPC DA só são compatíveis com aplicações baseadas em plataformas Microsoft.

Segundo Tan et al. (2007), devido às limitações do DCOM, a *OPC Foundation*, associação de indústrias sem fins lucrativos e que define os padrões OPC, criou um grupo técnico de trabalho para definir uma nova especificação denominada *OPC XML Data Access*. O OPC XML-DA provê uma melhor integração e interoperabilidade entre o chão de fábrica e os sistemas de monitoramento, manutenção e aplicações empresariais utilizando padrões adotados pela indústria, como XML, HTTP e SOAP (Figura 22).

Projetado para ter alto desempenho, o OPC XML-DA é compatível com multi-plataformas, apesar de o XML ser um dado textual e, com isso, poder causar um tráfego maior dos dados na rede. Por exemplo, para enviar um simples valor de um dado “0.555” o registro tem que estar bem formatado em XML para ser enviado, por exemplo, *<value xsi:type = “xsd:float”>0.555</value>*, significando que o tráfego na rede será seis vezes maior do que se tivéssemos que transmitir o valor original.

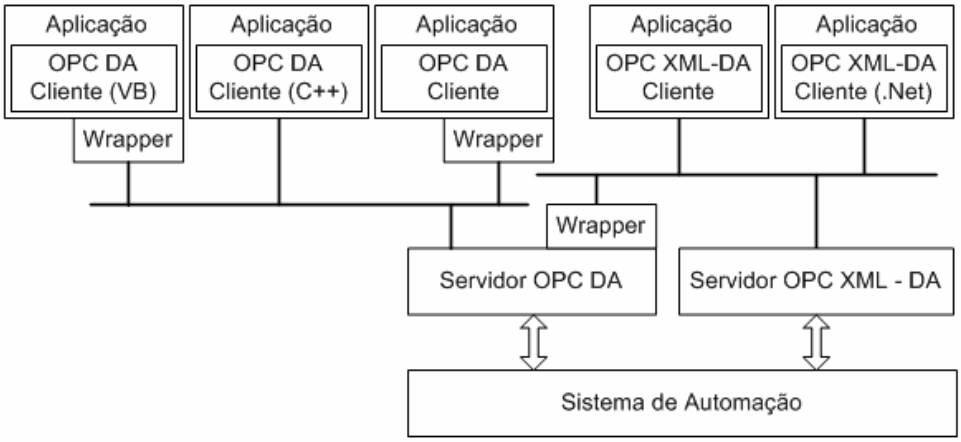

Figura 22 - *OPC DA* e *OPC XML-DA* provendo dados do chão de fábrica
Fonte: Tan et al. (2007)

Para então resolver o problema de compatibilidade entre a grande quantidade de servidores OPC DA com os novos servidores OPC XML-DA, Tan et al. (2007) propuseram criar em seu projeto o que eles chamaram de *Servidor DA Universal* para agregar múltiplos tipos de servidores OPC (Figura 23).

O *Servidor DA Universal* converte os dados recebidos em diferentes formatos dos servidores OPC para um único formato padrão antes de enviá-los para as camadas superiores. Ainda, o *Servidor DA Universal* permite configurar os direitos de acesso do cliente como acesso de somente leitura, somente escrita, ou leitura e escrita dos dados. Para Tan et al. (2007), a vantagem desta arquitetura de trabalho é permitir ao sistema um maior reuso, facilidade de manutenção, personalização e implantação em aplicações reais na indústria.

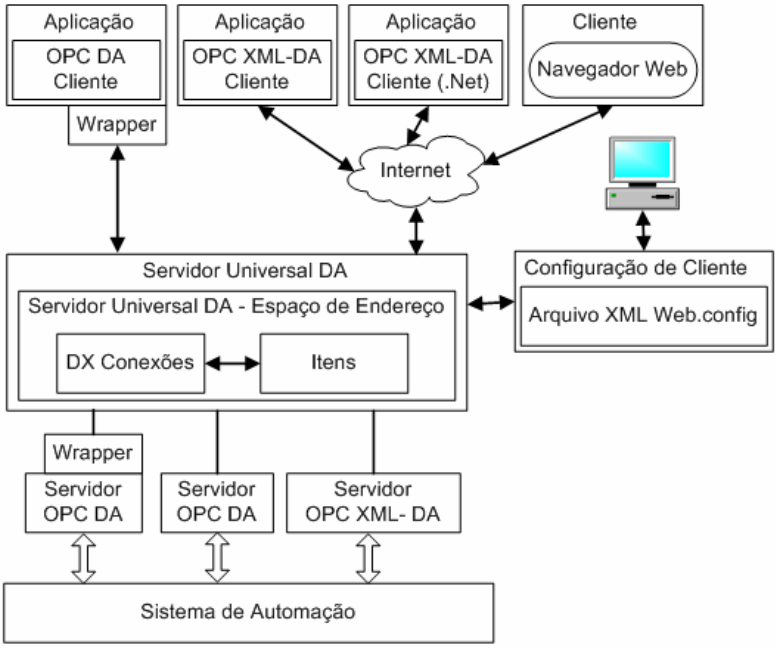

Figura 23 - Estrutura de aquisição e monitoramento de dados distribuídos baseado em OPC
Fonte: Tan et al. (2007)

Quanto ao desempenho, o projeto ainda tem algumas restrições com relação a sistemas em tempo real. A restrição está na dificuldade de encontrar um servidor OPC baseado em XML com tecnologia Internet que atualize os dados nos milisegundos necessários em aplicações em tempo real. Contudo, em aplicações do tipo *Sistema de Gestão Empresarial* (ERP) ou *Gerenciamento do Processo de Produção* (MES), onde a atualização dos dados pode ser na ordem de minutos, o projeto se mostrou satisfatório.

2.5.4 ESTRUTURA UTILIZANDO PORTAL DE SERVIÇOS

Para Kapsalis et al. (2002), a grande expansão das tecnologias de informação tem possibilitado a integração entre diversas aplicações. Esta evolução permite criar aplicações que podem ser acessadas do mundo todo. Algumas tecnologias têm sido introduzidas no ambiente industrial para prover estes tipos de aplicações, mas ainda não existe um padrão abrangente no setor industrial. Esta falta de padronização tem

conduzido à adoção de soluções voltadas para softwares de conectividade (*middleware*) a fim de integrar as diversas tecnologias embutidas nas aplicações do ambiente industrial. Dentre as três maiores tecnologias *middleware* (CORBA, DCOM e JAVA RMI), a tecnologia DCOM é a mais indicada para ser um padrão industrial.

Kapsalis et al. (2002) apresentam uma estrutura para integrar aplicações industriais utilizando a tecnologia DCOM. A Internet foi usada como ponto de acesso a essa estrutura, e o nome *Web Gateway* foi dado a este ponto de acesso via Internet. Através do *Web Gateway*, os usuários Web poderiam acessar serviços disponibilizados na indústria. O protocolo DCOM permite a chamada de componentes entre computadores, mas quando há um firewall neste caminho, o DCOM se mostra pouco amigável, por isso para contornar este problema, foi utilizado conjuntamente o protocolo HTTP.

Essa arquitetura para a aquisição de dados dos equipamentos de campo se assemelha a de Chau e Khai (2007), porém na abordagem de Kapsalis et al. (2002) utilizou-se uma rede de controle *Fieldbus* para conectar os equipamentos de campo, e a tecnologia OPC foi utilizada para providenciar uma entrada/saída universal. O DCOM foi utilizado como meio de conectividade entre a aplicação local e a LAN, e o protocolo HTTP foi utilizado para conectar as requisições do cliente na Internet com a LAN, conforme mostrado nas Figuras 24 e 25.

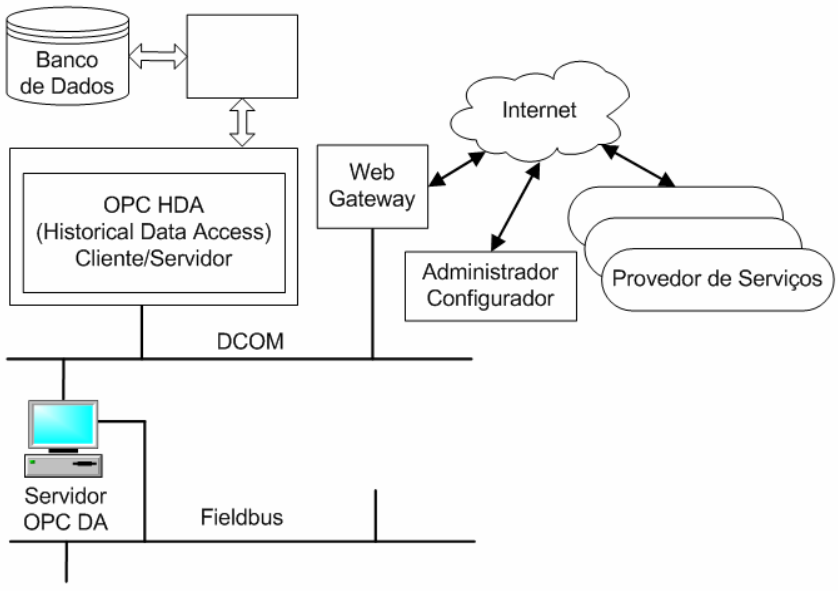

Figura 24 - Arquitetura do sistema
Fonte: Kapsalis et al. (2002)

Após a instalação do hardware, o administrador do sistema cria para cada usuário do sistema um diretório no qual serão criados grupos/itens do OPC DA e OPC HDA que o usuário terá acesso, como também o formato de visualização destes dados.

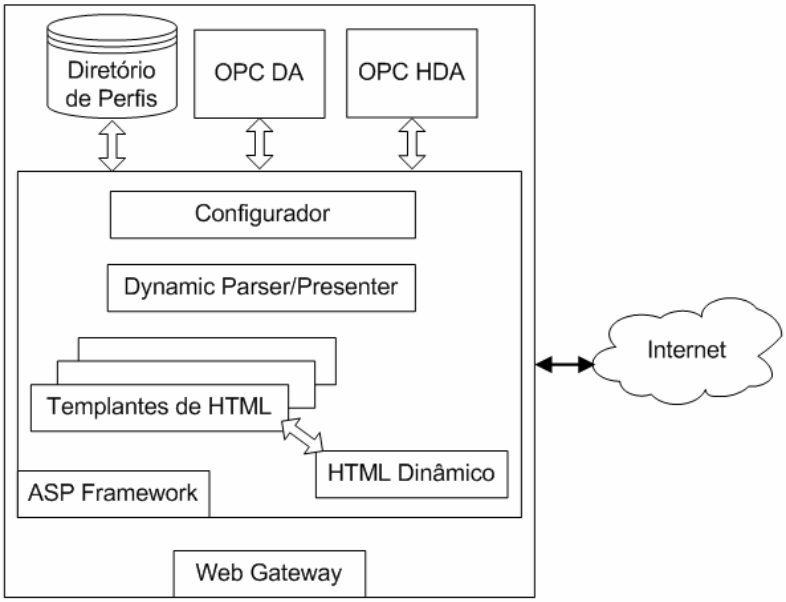

Figura 25 - O *Web Gateway*
Fonte: Kapsalis et al. (2002)

2.5.5 ESTRUTURA UTILIZANDO WEB SERVICES: PROPOSTA 1

Para Souza et al. (2006), desde o início da automação industrial, sempre houve uma grande necessidade de aprimorar a comunicação de dados entre os instrumentos e os controladores. A comunicação entre os dispositivos teve uma grande evolução com o aprimoramento das redes de comunicação. Este aprimoramento teve como conseqüência um grande aumento no número de processos de automação e equipamentos interligados através das aplicações distribuídas nas áreas de automação industrial. Toda essa evolução trouxe um grande desafio na padronização da comunicação interprocessos devido à falta de padronização nos protocolos de comunicação, mas com a criação do padrão OPC, eliminou-se as barreiras de comunicação impostas pela falta de padronização dos fabricantes (Figura 26).

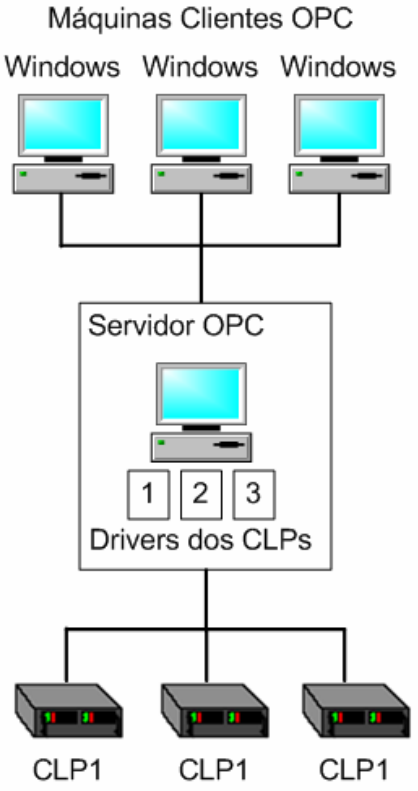

Figura 26 - Comunicação usando o padrão OPC
Fonte: Souza et al. (2006)

Para os pesquisadores da área de automação, o próximo passo seria criar novas alternativas para a integração de processos de automação ou aplicações corporativas já que as tecnologias utilizadas atualmente, como DCOM[20], CORBA, JAVA RMI, apresentam algum tipo de deficiência.

Com o surgimento da *Arquitetura Orientada a Serviços* (SOA), houve uma mudança de paradigma de tecnologias orientadas a objeto para tecnologias orientadas a serviços. Surgem então os *Web Services* como solução para uma melhor comunicação entre sistemas distribuídos (SOUZA et al., 2006).

Com a tecnologia *Web Services* é possível reunir processos que não foram originalmente construídos para trabalharem juntos ou transformar uma simples aplicação em um componente de software, permitindo assim a interoperabilidade com outros sistemas ou outras tecnologias (Figura 27).

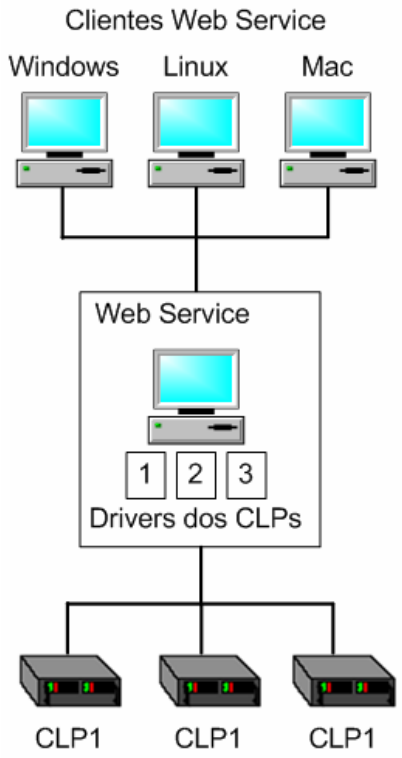

Figura 27 - *Web Service* encapsulando *drivers* de CLP
Fonte: Souza et al. (2006)

[20] DCOM (*Distributed Component Object Model*) é uma tecnologia proprietária da Microsoft® para criação de componentes de software distribuídos em computadores interligados em rede.

Com o objetivo de propor uma abordagem orientada a serviços para possibilitar o gerenciamento das informações de processos industriais, Souza et al. (2006) sugeriram implementar uma comunicação utilizando *Web Services* (Figura 28), então um *Web Service* utilizando Java e *drivers* de acesso ao CLP foi criado.

Entre o *Web Service* e o CLP, uma DLL foi desenvolvida para encapsular as funções de leitura e escrita ao CLP. Uma planta didática (Figura 29), com um reservatório de água montado, teve seu nível controlado por um CLP SIEMENS tipo S7-200, que recebia um sinal 4-20mA em uma de suas entradas analógicas, e um transmissor de pressão para o controle de nível.

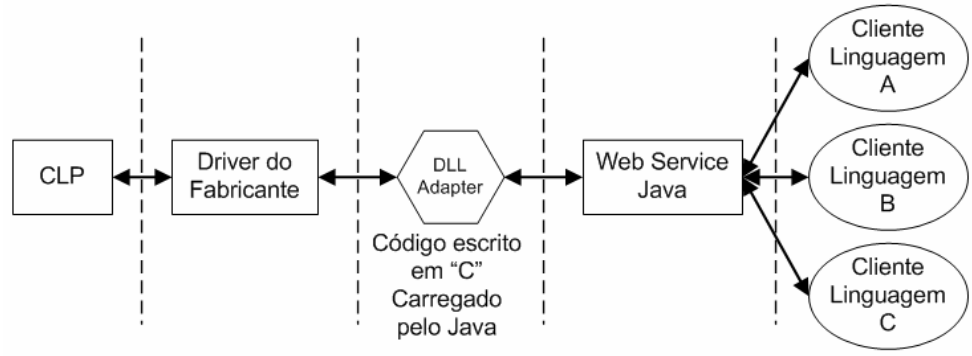

Figura 28 - Visão detalhada da arquitetura
Fonte: Souza et al. (2006)

Souza et al. (2006) criaram uma aplicação cliente e a distribuíram em cerca de trinta máquinas através de uma rede Ethernet. Ao inicializar a planta e o *Web Service*, as aplicações clientes passaram a obter os dados do CLP através de acessos diversos na rede e mesmo assim as respostas às solicitações dos clientes ficaram em torno de 200ms. Esta arquitetura possibilitou a flexibilidade das aplicações clientes, que poderiam utilizar qualquer sistema operacional ou serem desenvolvidas nas mais diversas linguagens de programação.

Concluiu-se então que são inúmeras as vantagens proporcionadas pela arquitetura orientada a serviços no que diz respeito às possibilidades de acesso a dados do chão de fábrica e distribuição dos dados aos setores de uma organização.

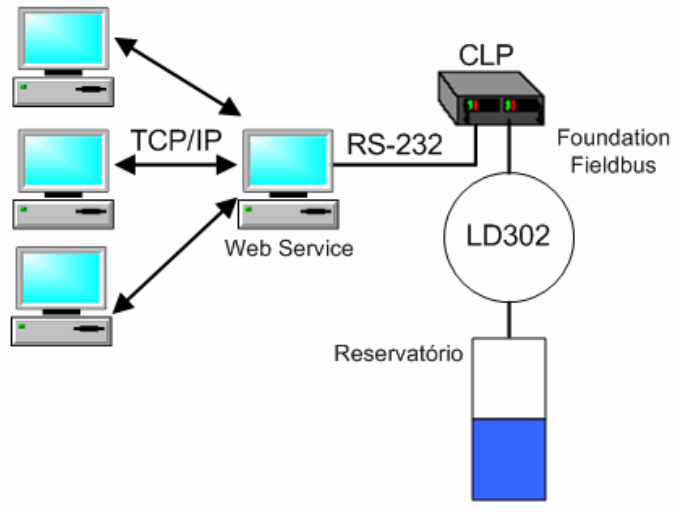

Figura 29 - Visão macro da arquitetura
Fonte: Souza et al. (2006)

2.5.6 ESTRUTURA UTILIZANDO WEB SERVICES: PROPOSTA 2

Segundo Kapsalis et al. (2003), as aplicações industriais, que compõem uma grande fonte de dados, ainda não adotaram uma arquitetura de serviços Web para disponibilizar todas essas informações para os usuários da indústria. Diante disso, Kapsalis et al. (2003) propuseram uma estrutura que chamaram de *e-services*. O objetivo do *e-services* é permitir uma provisão em larga escala de serviços Web a usuários finais. A principal tecnologia utilizada no *e-services* baseia-se em *Web Services*.

Os *Web Services* são uma evolução natural da Web para facilitar e viabilizar a interoperabilidade entre aplicações. Utilizando *Web Services*, há uma diminuição da complexidade das regras de negócio nas interações entre as aplicações. Os *Web Services*

possibilitam um baixo acoplamento entre os softwares, além de trafegarem sobre a Internet utilizando as tecnologias mencionadas anteriormente (HTTP, XML e SOAP).

Duas outras tecnologias foram utilizadas: OPC e DCOM. A tecnologia OPC, baseada no DCOM, permitiu a integração das aplicações industriais e o compartilhamento de dados através da LAN. Para Kapsalis et al. (2003), apesar de melhorias feitas no DCOM no passar dos anos, esta tecnologia não é ideal para todos os casos, e suas principais limitações seriam as seguintes:

- A tecnologia é mais bem suportada em plataformas Microsoft®;
- O protocolo DCOM não foi projetado levando em consideração a Internet;
- A geração de mensagens de comunicação que usam o protocolo COM; mensagens enviadas pelo COM sobre a Internet esbarram em problemas principalmente devido a firewalls.

Kapsalis et al. (2003) definem um *Provedor de Serviços* como "uma entidade que provê algum tipo de *e-service* que requer uma conexão entre a entidade e um cliente da automação industrial". O cliente da automação, neste caso, pode ser tanto um usuário final quanto aplicações do tipo caldeira, filtro, sistema de alarmes, etc, que requerem informações dos equipamentos instalados na rede de automação. Duas estruturas de conexão entre *Provedores de Serviços* e clientes da rede de automação são citados. A primeira estrutura é denominada Ponto-A-Ponto (Figura 30).

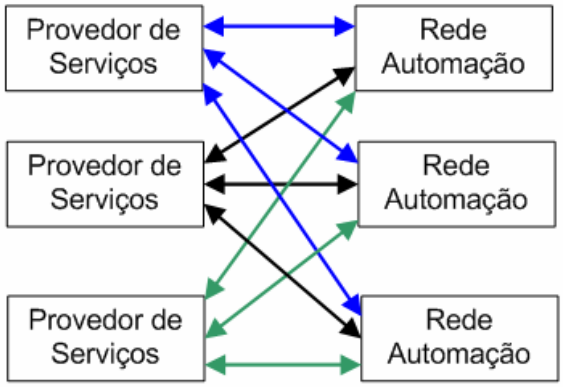

Figura 30 - Estrutura Ponto-A-Ponto
Fonte: Kapsalis et al. (2003)

Na abordagem Ponto-A-Ponto, o *Provedor de Serviços n* tem que manter uma conexão de rede separada para cada cliente *m* da rede de automação. Este tipo de estrutura é bem flexível e, no caso de poucas conexões, o gerenciamento não é complexo. A partir do momento que novas conexões *n* x *m* vão sendo adicionadas, o gerenciamento de todas as conexões começa a se tornar complexo.

A segunda estrutura é denominada *Hub-And-Spoke* (Figura 31). Na estrutura *Hub-And-Spoke*, adiciona-se um ponto central chamado *Service Aggregator* entre os clientes da rede de automação e os *Provedores de Serviços*.

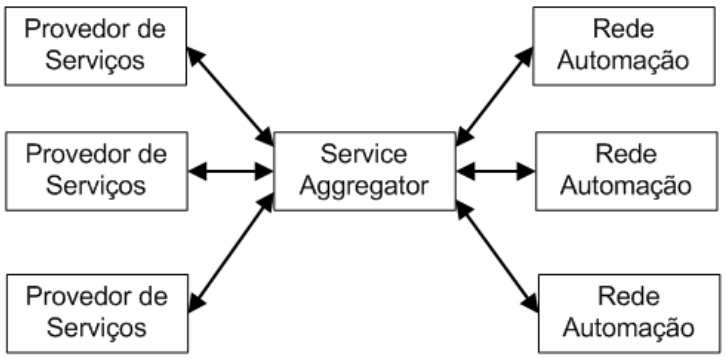

Figura 31 - Estrutura *Hub-And-Spoke*
Fonte: Kapsalis et al. (2003)

Comparando a estrutura *Hub-And-Spoke* com a Ponto-A-Ponto, onde o número de conexões é da ordem de *nxm*, o número de conexões entre os *Provedores de Serviços*

n e os clientes *m* é da ordem de *n+m*, assim a estrutura *Hub-And-Spoke* é uma solução mais vantajosa ao longo do tempo em termos de escalabilidade. Por isso, Kapsalis et al. (2003) utilizavam a estrutura *Hub-And-Spoke* em sua abordagem conforme a Figura 32. No centro da figura temos o *Service Aggregator* fazendo a ligação entre os clientes da rede de automação do lado direito com os sistemas de serviços do lado esquerdo.

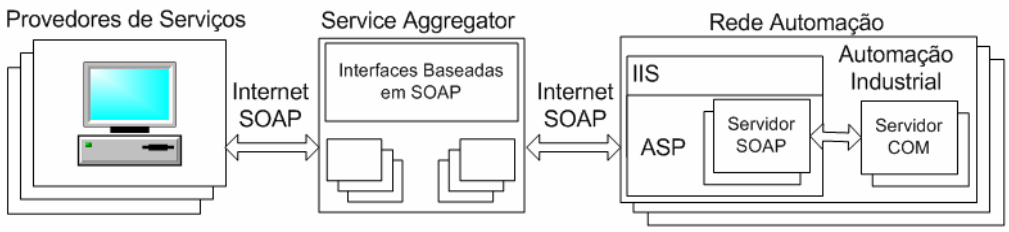

Figura 32 - *Service Aggregator*
Fonte: Kapsalis et al. (2003)

O *Service Aggregator*, além de prover acesso a serviços expostos por servidores SOAP remotos localizados em automações remotas de clientes, também executa serviços como autenticação e encriptação, controle de acesso, agendamento, entre outros (Figura 33).

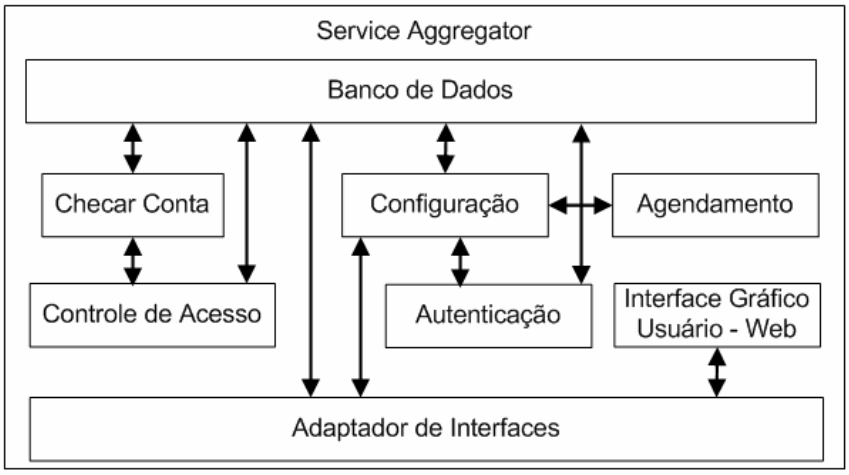

Figura 33 - Bloco de funções do *Service Aggregator*
Fonte: Kapsalis et al. (2003)

3 METODOLOGIA

Este capítulo discute as principais estruturas utilizadas na aquisição de dados via Internet, aborda os possíveis problemas em cada uma e propõe uma estrutura para minimizar esses problemas.

3.1 ESTRUTURAS MAIS UTILIZADAS NA OBTENÇÃO DE DADOS VIA INTERNET

Analisando as estruturas do capítulo dois para obtenção de dados de plantas industriais, através da Internet, tem se uma estrutura básica mostrada na Figura 34. Esta estrutura solicita informações de dispositivo de campo através de um navegador Web. Através de uma página da Internet executada no navegador Web, uma solicitação é feita ao servidor Web localizado no ambiente industrial. Este servidor Web, por sua vez, efetua a solicitação ao Servidor OPC para obter os dados do dispositivo de campo conectado ao barramento Foundation Fieldbus. O servidor Web transforma os dados obtidos em formato HTML e envia para o navegador Web do cliente.

O inconveniente deste tipo de estrutura refere-se à atualização do navegador Web com as novas informações que estão sendo geradas no ambiente industrial. Para

obter estas atualizações, o navegador Web deve efetuar solicitações constantemente ao servidor Web. Em geral, este procedimento é feito manualmente pelo usuário.

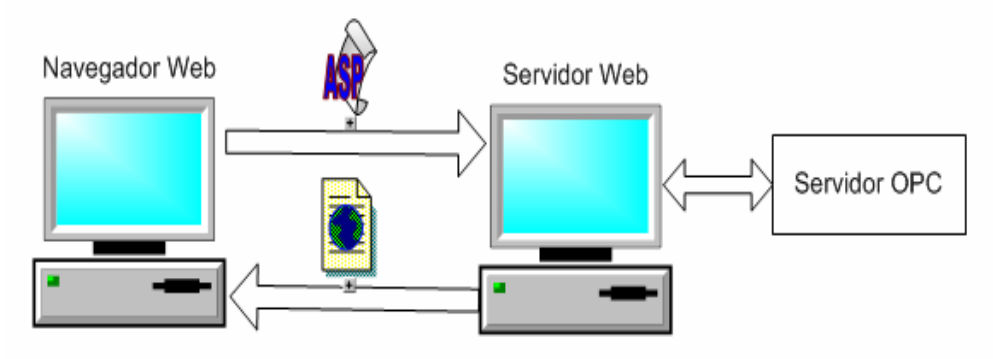

Figura 34 - Estrutura para aquisição de dados via Internet

Para contornar este inconveniente, alguns mecanismos de atualização automática de páginas são adotados, permitindo um melhor desempenho na obtenção dos dados. Um desses mecanismos é o AJAX. O AJAX não é uma tecnologia, mas sim o uso em conjunto de várias tecnologias providas por navegadores, como o *Javascript* e XML. Utilizando o AJAX, as solicitações são feitas automaticamente entre o cliente e o servidor (Figura 35). Estas solicitações automáticas são feitas de maneira assíncrona, aumentando a interatividade do usuário final com as páginas.

Outra dificuldade encontrada nesse tipo de estrutura aparece quando há necessidade de armazenar as informações recebidas pelo navegador Web, ou seja, páginas Web no formato HTML. Em HTML, as informações recebidas e os comandos para formatá-las na tela estão todos juntos no mesmo arquivo. Esta característica dificulta o armazenamento dos dados recebidos.

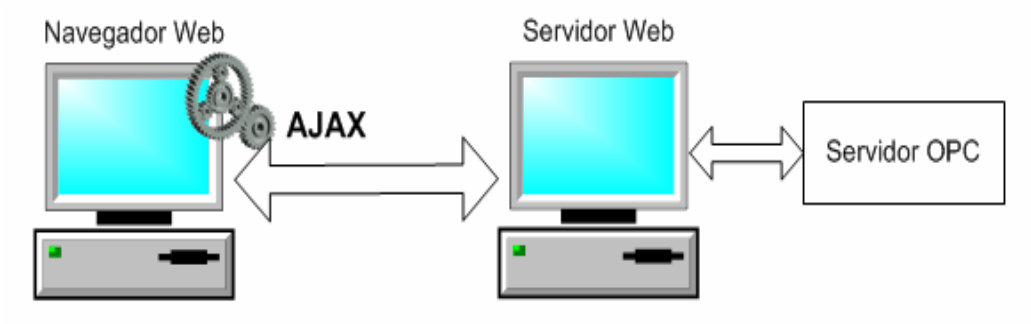

Figura 35 - Atualização de dados utilizando AJAX

Para solucionar o problema de armazenamento dos dados do lado do cliente, uma das estruturas mais adotadas tem sido a de *Web Services*. Os *Web Services*, como mencionado no capítulo dois, disponibilizam a comunicação entre aplicações através de serviços oferecidos por componentes de um sistema distribuído. Esta característica permite que uma aplicação cliente solicite dados do ambiente industrial a um servidor Web, e receba como resposta os dados formatados em XML. Desta forma, os dados recebidos do lado do cliente são facilmente manipulados e armazenados.

A Figura 36 ilustra uma estrutura padrão utilizando *Web Services* como forma de obtenção de dados de planta industrial. Analisando este tipo de estrutura pode-se verificar que há um problema de desempenho quando a aplicação cliente efetua solicitações repetidamente a um servidor Web, pelo modo como as chamadas são feitas aos *Web Services*. Essas chamadas podem ser de dois tipos: síncronas ou assíncronas.

Figura 36 - Estrutura para aquisição de dados através de *Web Services*

O modo de chamada do tipo "Síncrono" causa a paralisação da aplicação cliente enquanto o *Web Service* está processando a chamada (Figura 37). O tempo de paralisação irá depender de vários fatores, entre eles, da quantidade de dados solicitados, da velocidade de processamento das máquinas envolvidas no processo, do tipo de conexão Web, etc.

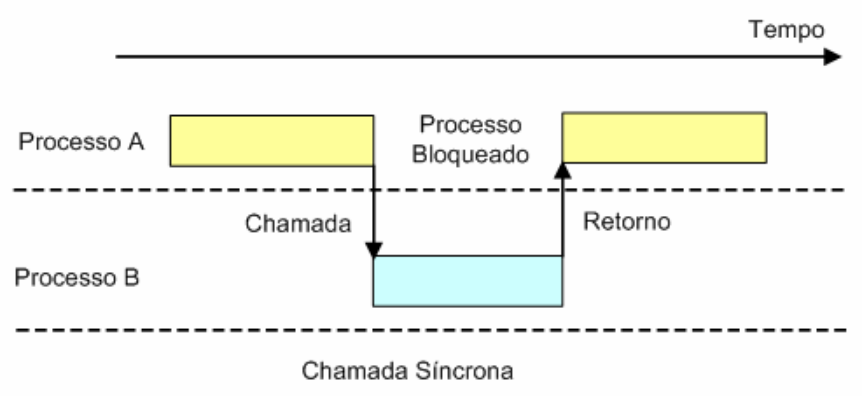

Figura 37 - Chamada de *Web Service* no modo síncrono

O segundo modo de chamada, o modo "Assíncrono", permite que a aplicação cliente continue trabalhando enquanto o *Web Service* processa a chamada (Figura 38). Outra vantagem do modo assíncrono é em relação ao tempo de processamento. Este tempo é percebido quando há uma grande quantidade de solicitações a serem executadas. Por exemplo, se dez grupos de valores diferentes devem ser obtidos de um dispositivo de campo, para cada grupo será necessário efetuar uma chamada ao *Web Service*. Supondo que cada chamada demore três segundos para ser executada, usando-

se a chamada síncrona (uma chamada esperando a outra), o tempo total será de trinta segundos. Se a chamada assíncrona ("todas ao mesmo tempo") for utilizada, o tempo total será de poucos segundos.

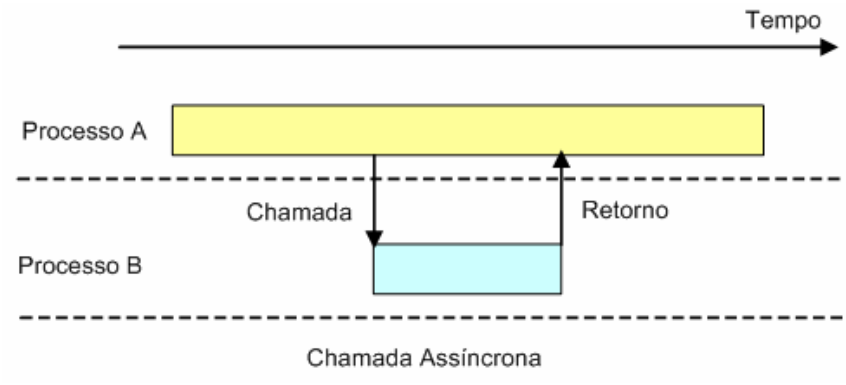

Figura 38 - Chamada de *Web Service* no modo assíncrono

Mesmo com a otimização proporcionada pelo modelo de chamada assíncrona, existem algumas situações em que existe perda de recurso, quando a aplicação cliente está solicitando repetidamente o valor de uma variável que quase não se altera. Pode-se observar esta situação quando, por exemplo, uma aplicação cliente está monitorando a temperatura de um freezer. A aplicação cliente é configurada para chamar o *Web Service* a cada um minuto para obter o valor da temperatura, mas a temperatura do freezer permanece inalterada durante dias antes de abaixar meio grau.

Durante todo esse período em que não houve alteração da temperatura, a aplicação cliente esteve chamando o *Web Service* para obter o valor e armazená-lo. Isto acarreta uma grande perda de recurso, sem contar o tráfego de informações desnecessárias entre a aplicação cliente, o Servidor Web e o Servidor OPC.

3.2 PROPOSTA DE UMA ESTRUTURA IDEAL PARA OBTENÇÃO DE DADOS VIA INTERNET

3.2.1 VISÃO GERAL

A estrutura proposta como ideal é obtida criando-se dois módulos: o cliente e o servidor. O módulo cliente é instalado na máquina cliente, e o módulo servidor no ambiente industrial. O módulo cliente envia para o módulo servidor, via Internet, as variáveis que deseja monitorar e, após o envio, o módulo cliente não deve ficar travado esperando o retorno do módulo servidor. O módulo servidor recebe a solicitação e passa a monitorar as variáveis no ambiente industrial. O módulo servidor somente irá enviar dados para o módulo cliente quando algum valor de uma das variáveis for alterado.

3.2.2 FORMA DE IMPLEMENTAÇÃO DA ESTRUTURA PROPOSTA

A estrutura proposta pode ser implementada utilizando os recursos disponíveis nos *Web Services*, no sistema de eventos do Windows e no Servidor OPC. A Figura 39 exibe como os módulos são montados no ambiente do cliente e do servidor, e também o fluxo das informações entre os módulos. O módulo cliente é composto por três componentes: aplicativo cliente, sistema de eventos e *Web Service*. O módulo servidor é composto por quatro componentes: aplicativo servidor, sistema de eventos, *Web Service* e Servidor OPC.

O ponto chave desta estrutura está na comunicação entre os *Web Services* e os aplicativos cliente ou servidor, porque os *Web Services* não têm como se comunicarem diretamente com um aplicativo executável. Vários mecanismos podem ser utilizados como alternativa para efetuar esta comunicação, tais como sistema de arquivos, banco de dados, etc. Observando as estruturas analisadas no capítulo dois, não foi encontrada nenhuma que utilizasse o sistema de eventos do Windows. Apesar dos eventos apresentarem certa complexidade de implementação, eles oferecem um melhor desempenho na comunicação entre quem envia e quem recebe os eventos.

A obtenção de dados se inicia com a aplicação cliente passando as variáveis que se deseja monitorar para o *Web Service* instalado no ambiente industrial, conforme ilustrado no item 1 da Figura 39. O *Web Service*, por sua vez, repassa as variáveis para o aplicativo servidor através do sistema de eventos (item 2). O aplicativo servidor configura então o Servidor OPC para monitorar as variáveis (item 3). Quando alguma variável tem seu valor alterado, o Servidor OPC envia um evento, chamado *Data Change*, com o novo valor para o aplicativo servidor (item 4). O evento *Data Change* é enviado somente quando alguma variável é alterada na planta industrial.

O aplicativo servidor envia os novos valores para o módulo cliente através do *Web Service* cliente (item 5). O *Web Service* repassa os valores para o aplicativo cliente através do sistema de eventos (item 6). De posse dos valores das variáveis, o aplicativo cliente efetua o armazenamento das mesmas em um banco de dados (item 7), e exibe as variáveis para o usuário final. Uma vez que as variáveis estejam armazenadas no banco de dados, será possível criar inúmeros relatórios para observar seu comportamento, e analisando este comportamento pode-se predizer o mau funcionamento de um equipamento ou quando ele está preste a quebrar.

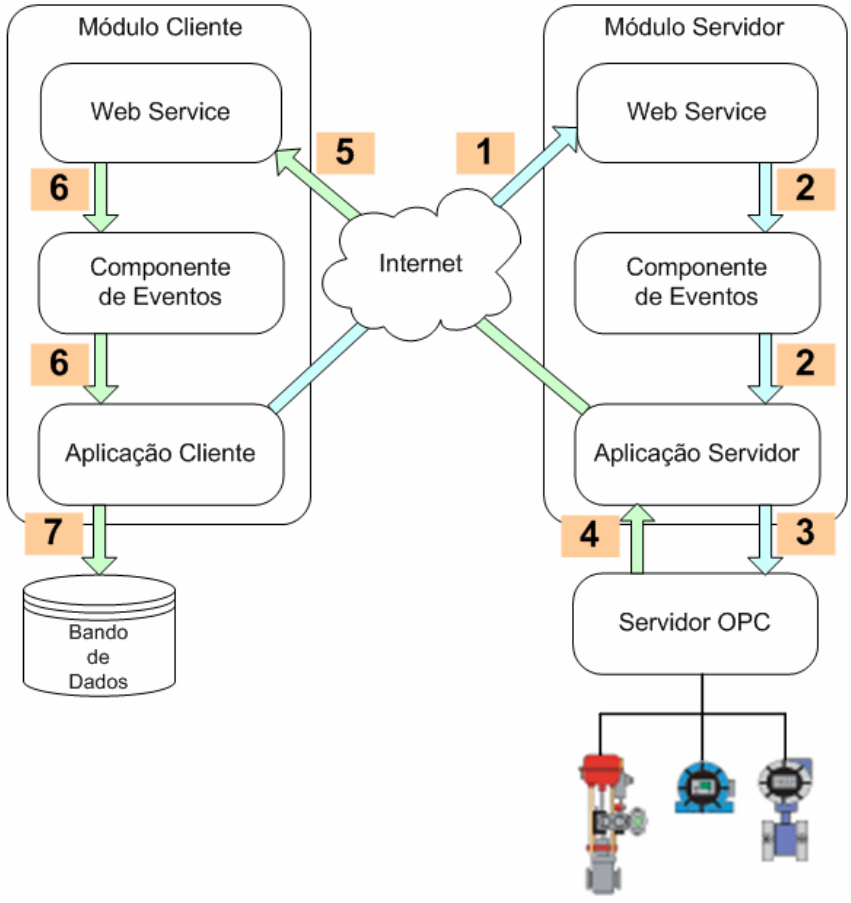

Figura 39 - Estrutura proposta para aquisição de dados via Internet

3.3 FASE DO DESENVOLVIMENTO

A fase de desenvolvimento tem como objetivo levantar os requisitos e as soluções encontradas para o desenvolvimento do sistema de aquisição de dados de dispositivos de campo via Internet. Esta seção descreve a execução do projeto do protótipo; as principais arquiteturas, estruturas de dados; a codificação do protótipo e a implantação do projeto.

3.3.1 MATERIAIS E MÉTODOS

Esta subseção descreve as ferramentas utilizadas no projeto e desenvolvimento do software deste trabalho.

No desenvolvimento dos diagramas UML utilizados para criar o projeto, bem como na modelagem do banco de dados, foi utilizada a ferramenta *Microsoft Visio® for Enterprise Architects (10.0.2705)*. Para o desenvolvimento do software, foi utilizado o *Visual Studio® 2005* da Microsoft®. Quando esta versão do *Visual Studio* é instalada, ela instala automaticamente o *Microsoft .NET Framework versão 2.0*. A instalação é feita porque tanto a utilização do *Visual Studio* como a execução das aplicações desenvolvidas nele depende da versão 2.0 do *Microsoft .NET Framework*. Um fator importante para a escolha do *Microsoft Visio* para a modelagem do sistema foi sua integração com o *Visual Studio*. Esta integração permite que as classes desenhadas no *Visio* sejam transformadas em código dentro do *Visual Studio*. O *Visio* permite também a modelagem de banco de dados e, a partir dessa modelagem, a geração de *Scripts* para a criação do banco de dados.

Entre as opções de linguagem de programação disponíveis no *Visual Studio .NET*, foi escolhida a linguagem *Visual Basic* para a geração do código do protótipo. Para armazenamento dos dados, foi utilizado o *SGBD SQL Server 2000* da Microsoft utilizando o *Enterprise Manager* para a criação do bando de dados e das tabelas.

3.3.2 ARQUITETURA PRELIMINAR

O projeto será dividido em dois módulos, Módulo Cliente e Módulo Servidor, que serão divididos em camadas lógicas de software. Esta divisão auxilia a diminuir a

complexidade do desenvolvimento e permite uma maior flexibilidade no caso de mudanças futuras. Assim, a solução adotada foi o desenvolvimento em três camadas: Camada de Apresentação, Camada de Regra de Negócios e Camada de Acesso a Dados (Figura 40).

Camada de Apresentação (Interface)
Camada de Regra de Negócios (BUS)
Camada de Acesso a Dados (DB)

Figura 40 - Camadas lógicas de software

A Camada de Apresentação é onde será feita toda a interação entre o usuário e o sistema. A Camada de Regra de Negócios será responsável pela execução das solicitações da Camada de Apresentação, e aplicará regras específicas para cada um dos tipos de transações que ocorrem com as informações do sistema. A Camada de Acesso a Dados será responsável por efetuar as operações com o banco de dados.

Esta abordagem em três camadas possibilita que, no futuro, possam-se trocar os componentes de uma camada sem afetar as demais.

3.3.3 TABELA DE ATORES

O objetivo do ator é interagir com o sistema. Normalmente os atores são considerados seres humanos, mas também podem ser outros sistemas, temporizadores ou dispositivos de hardware. Os atores que interagem com o sistema foram identificados na Tabela 03.

Tabela 03 - Atores Propostos para o Sistema

Ator	Definição
Usuário Cliente	Faz a manutenção das informações necessárias para o funcionamento do Módulo Cliente e interage para o monitoramento de Tags.
Módulo Cliente	Interage com o Módulo Servidor, enviando solicitações e processando as respostas recebidas.
Temporizador Monitoramento	Faz a atualização visual das informações que estão sendo monitoradas.
Usuário Servidor	Opera o Módulo Servidor.
Módulo Servidor	Interage com o Módulo Cliente, retornando as respostas das solicitações recebidas.
Temporizador Ping	Envia sinais do Módulo Servidor para o Módulo Cliente para indicar que a comunicação entre os módulos está funcionando corretamente.
Servidor OPC	Interage com o Módulo Servidor, retornando as informações que foram solicitadas.

3.3.4 TABELA DE EVENTOS

A função da tabela de eventos é representar os estímulos recebidos pelo sistema, originados tanto externamente como internamente, e permitir que as interações do sistema sejam identificadas rapidamente. Estímulos externos são ações recebidas pelo sistema e provocadas por um usuário ou outro aplicativo (Atores). Estímulos internos normalmente são provocados por algum processo independente interagindo com o sistema, como por exemplo, um temporizador.

A Tabela 04 mostra os eventos recebidos pelo sistema, que seguem o formato *Resposta = Sujeito + Verbo + Objeto*, onde:

- Sujeito: é um ator identificado anteriormente na tabela de atores;
- Verbo: indica a ação efetuada pelo sujeito;
- Objeto: é o foco da ação definida no verbo;
- Resposta: o resultado final da ação.

Tabela 04 - Eventos Propostos para o Sistema

No.	Sujeito	Verbo	Objeto	Resposta
01	Usuário Cliente	Iniciar	Módulo Cliente	Checagem do funcionamento do *Web Service* Cliente e do sistema de eventos.
02	Usuário Servidor	Iniciar	Módulo Servidor	Checagem do funcionamento do *Web Service* Servidor e sistema de eventos.
03	Usuário Cliente	Inserir	Local	Cadastro de informações no sistema sobre local remoto de onde serão efetuadas aquisições de dados.
04	Usuário Cliente	Pesquisar	Local	Exibe informações sobre locais remotos cadastrados no sistema.
05	Usuário Cliente	Alterar	Local	Alteração de informações no sistema sobre local remoto.
06	Usuário Cliente	Excluir	Local	Exclui informações sobre local remoto.
07	Usuário Cliente	Inserir	Dispositivo	Cadastro de informações no sistema sobre dispositivos de campo.
08	Usuário Cliente	Pesquisar	Dispositivo	Exibe informações sobre dispositivos de campo cadastrados no sistema.
09	Usuário Cliente	Alterar	Dispositivo	Alteração de informações no sistema sobre dispositivos de campo.
10	Usuário Cliente	Excluir	Dispositivo	Exclui informações sobre os dispositivos de campo.
11	Usuário Cliente	Inserir	Tag	Cadastro de informações no sistema sobre tags de dispositivos de campo.
12	Usuário Cliente	Pesquisar	Tag	Exibe informações de tags de dispositivos de campo cadastrados no sistema.
13	Usuário Cliente	Alterar	Tag	Alteração de informações no sistema sobre tags de dispositivos de campo.
14	Usuário Cliente	Excluir	Tag	Exclui informações sobre tags de dispositivos de campo.
15	Usuário Cliente	Selecionar Dados de Monitoramento	Módulo Cliente: Monitoramento	Seleção de informações de cadastros efetuados anteriormente, que são adicionadas na parte de pesquisa para serem utilizadas no monitoramento de tags.
16	Usuário Cliente	Iniciar Monitoramento	Módulo Cliente: Monitoramento	Envio de tags a serem monitoradas, para o Módulo Servidor. Inicializa o Temporizador Monitoramento.
17	Módulo Cliente	Inicializar	Temporizador Monitoramento	Início de um ciclo infinito de busca e exibição de dados atualizados do monitoramento de tags.

No.	Sujeito	Verbo	Objeto	Resposta
18	Módulo Servidor	Receber Iniciar Monitoramento	Módulo Servidor: Monitoramento	Inclusão de tags a serem monitoradas no Servidor OPC. Retorna o identificador *GroupHandle* fornecido pelo Servidor OPC para o Módulo Cliente, relativo ao grupo de tags que estão sendo monitoradas. Inicializa o Temporizador Ping.
19	Módulo Servidor	Inicializar	Temporizador Ping	Início de um ciclo infinito que envia sinais para o Módulo Cliente, indicando que a comunicação está funcionando corretamente.
20	Módulo Cliente	Receber Iniciar Monitoramento	Módulo Cliente: Monitoramento	Atualização dos dados de monitoramento com *GroupHandle* recebido do Módulo Servidor.
21	Servidor OPC	Enviar Alteração de Dados	Módulo Servidor	Envio de valores de tags monitoradas, após recebimento do evento *DataChange* no Módulo Servidor.
22	Módulo Servidor	Enviar Alteração de Dados	Módulo Cliente	Atualização dos dados de monitoramento com valores recebidos das tags.
23	Temporizador Monitoramento	Atualizar Exibição de Dados	Módulo Cliente	Exibe os dados atualizados do monitoramento de tags.
24	Temporizador Ping	Enviar Sinal Ping	Módulo Cliente	Recebimento e processamento do sinal Ping.
25	Usuário Cliente	Cancelar Monitoramento	Módulo Servidor	Cancelamento de monitoramento, enviado para o Módulo Servidor. Pára o Temporizador Monitoramento.
26	Temporizador Monitoramento	Parar	Módulo Cliente: Monitoramento	Pára o ciclo infinito de busca e exibição de dados.
27	Módulo Servidor	Cancelar Monitoramento	Servidor OPC	Remoção de tags que estavam sendo monitoradas pelo Servidor OPC. Pára o Temporizador Ping.
28	Temporizador Ping	Parar	Módulo Servidor	Pára o ciclo infinito de envio de Ping.

3.3.5 CASOS DE USO

Os casos de uso são documentos que descrevem o quê o sistema deverá fazer, e não como ele o fará. Os casos de usos descritos a seguir foram elaborados para satisfazer todos os eventos identificados na tabela de eventos. Um caso de uso pode ser responsável por satisfazer um ou mais eventos. Assim, analisando-se a tabela de

eventos, nota-se que alguns eventos podem ser agrupados em um caso de uso, como, por exemplo, os que tratam de inserir, pesquisar, alterar e excluir tags. A Tabela 05 mostra os agrupamentos de eventos e os casos de uso propostos para satisfazê-los.

Tabela 05 - Agrupamento de Casos de Uso vs. Eventos

Caso de Uso	Número do Evento
Iniciar Módulo Cliente	01
Iniciar Módulo Servidor	02
Manter Local	03, 04, 05, 06
Manter Dispositivo	07, 08, 09, 10
Manter Tag	11, 12, 13, 14
Selecionar Monitoramento	15
Iniciar Monitoramento	16, 17, 18, 19, 20
Atualizar Dados Monitorados	21, 22
Pesquisar Dados Monitorados	23
Checar Comunicação	24
Cancelar Monitoramento	25, 26, 27, 28

3.3.6 DIAGRAMA DE CASOS DE USO

O diagrama de casos de uso descrito na Figura 41 foi elaborado considerando-se as Tabelas 04 e 05. Neste diagrama são mostrados os atores, os casos de uso e seus relacionamentos.

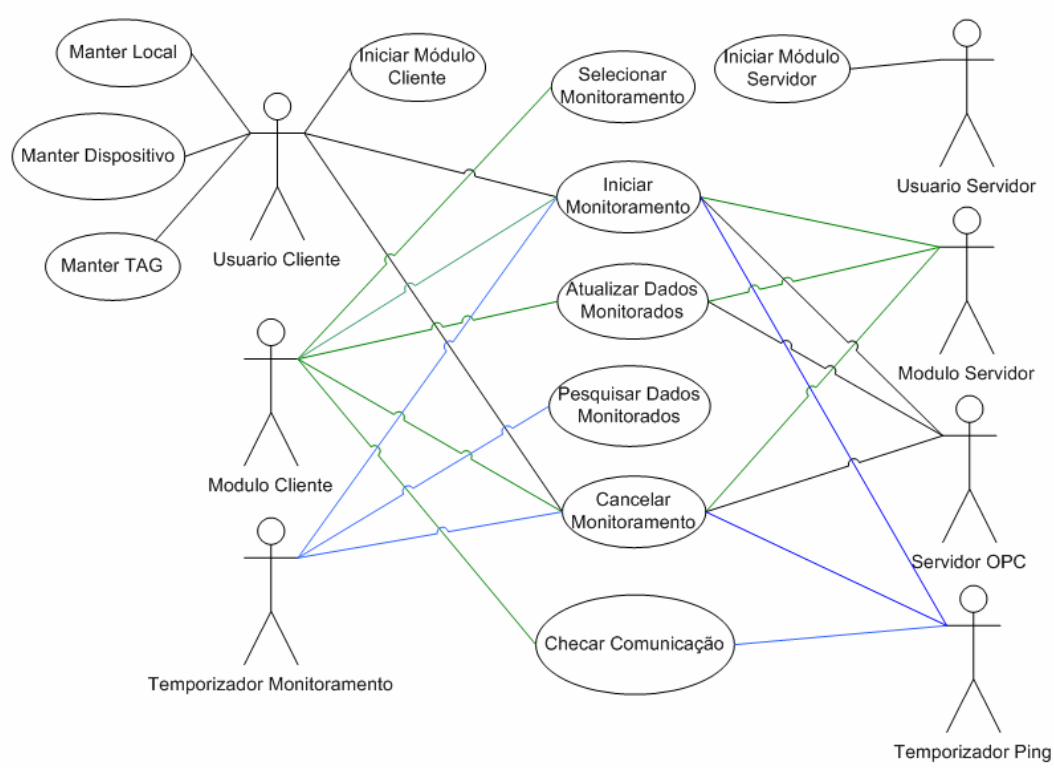

Figura 41 - Diagrama de casos de uso

3.3.7 DETALHAMENTO DOS CASOS DE USO

Os modelos de casos de uso existem para mostrar mais detalhes sobre os casos de uso e promover uma melhor compreensão sobre os processos e requisitos. Estes modelos fornecem mais informações sobre as interações simuladas por eventos e agora atribuídas aos casos de uso, e demonstram o que acontece quando um caso de uso responde a um evento. Nos modelos, foram identificados três níveis de caminhos, primário, alternativo e de exceção, que um caso de uso segue em resposta a um evento:

- Primário (*Happy Pathway*): é o caminho perfeito onde nada sai errado. Caso haja mais de um caminho primário, apenas um é escolhido como o caminho primário.
- Alternativo (*Alternate Pathway*): é um caminho considerado bom; ele apenas não é o mais utilizado.

- Exceção (*Exception Pathway*): destina-se a capturar as condições de erros que podem acontecer no sistema, ou caminhos contrários ao caminho primário.

3.3.8 CASO DE USO: INICIAR MÓDULO CLIENTE

Descrição:	Este caso de uso começa quando o Usuário Cliente inicia o Módulo Cliente. O objetivo deste caso de uso é efetuar todas as checagens necessárias para que o sistema funcione corretamente. Ele termina quando o Usuário Cliente fecha o Módulo Cliente.
Atores:	Usuário Cliente.
Pré-condições:	
Caminho Primário	
Descrição:	**Iniciar Módulo Cliente.**
Detalhe:	1 - Sistema verifica se *Web Service* local está funcionando corretamente. 2 - Sistema verifica se sistema de eventos está funcionando corretamente.
Caminhos Alternativos	
Descrição:	
Detalhe:	
Caminhos de Exceção	
Descrição:	***Web Service* ou sistema de eventos não está respondendo.**
Detalhe:	1 - Sistema informa que *Web Service* não está respondendo. 2 - Sistema informa que os eventos não estão respondendo.
Regras de Negócio	
Fato Estrutural:	
Ação Restritiva:	Caso o *Web Service* ou sistema de eventos não esteja respondendo, a operação de monitoramento deverá ser bloqueada.
Gatilho de Ação:	
Interface:	A interface deverá ter ícones informando o estado de operação do sistema de eventos e do *Web Service*, ou seja, se estão funcionando corretamente ou não.
Cálculo:	

3.3.9 CASO DE USO: INICIAR MÓDULO SERVIDOR

Descrição:	Este caso de uso começa quando o usuário inicia o Módulo Servidor. O objetivo deste caso de uso é efetuar todas as checagens necessárias para que o sistema funcione corretamente. Ele termina quando o Usuário Cliente fecha o Módulo Servidor.
Atores:	Usuário Servidor.
Pré-condições:	
Caminho Primário	
Descrição:	**Iniciar módulo servidor.**
Detalhe:	1 - Sistema verifica se *Web Service* local está funcionando corretamente. 2 - Sistema verifica se sistema de eventos está funcionando corretamente.
Caminhos Alternativos	
Descrição:	
Detalhe:	
Caminhos de Exceção	
Descrição:	**Web Service ou sistema de eventos não está respondendo.**
Detalhe:	1 - Sistema informa que o *Web Service* não está respondendo. 2 - Sistema informa que os eventos não estão respondendo.
Regras de Negócio	
Fato Estrutural:	
Ação Restritiva:	Caso o *Web Service* ou sistema de eventos não esteja respondendo, a operação de monitoramento não funcionará corretamente.
Gatilho de Ação:	
Interface:	A interface deverá ter ícones informando o estado de operação do sistema de eventos e do *Web Service*, ou seja, se estão funcionando corretamente ou não.
Cálculo:	

3.3.10 CASO DE USO: MANTER LOCAL

Descrição:	Este caso de uso começa quando o Usuário Cliente inicia o aplicativo que manipula as informações dos locais remotos. O objetivo deste caso de uso é efetuar as operações relacionadas à inserção, alteração, pesquisa e exclusão de informações de locais remotos. Ele termina quando o Usuário Cliente encerra o aplicativo de locais remotos.
Atores:	Usuário Cliente.
Pré-condições:	
Caminho Primário	
Descrição:	**Inserir local remoto.**
Detalhe:	1 - Ator informa os dados sobre local remoto e efetua a inserção. 2 - Sistema verifica se as informações já foram cadastradas anteriormente. 3 - Sistema armazena as informações. 4 - Sistema lista informações de locais remotos já cadastrados no sistema.
Caminhos Alternativos	
Descrição:	**Alterar local remoto.**
Detalhe:	1 - Ator seleciona local remoto cadastrado anteriormente. 2 - Ator altera informações e efetua a atualização. 3 - Sistema verifica se informações já foram cadastradas anteriormente. 4 - Sistema atualiza as informações. 5 - Sistema lista informações de locais remotos já cadastrados no sistema.
Descrição:	**Pesquisar local remoto.**
Detalhe:	1 - Ao abrir o aplicativo, ator deverá receber a listagem de locais remotos já cadastrados no sistema.
Caminhos de Exceção	
Descrição:	**Excluir local remoto.**
Detalhe:	1 - Ator seleciona local remoto cadastrado anteriormente. 2 - Ator efetua a exclusão. 3 - Sistema verifica se as informações já foram utilizadas anteriormente no sistema. 4 - Sistema exclui as informações. 5 - Sistema lista informações de locais remotos já cadastrados no sistema.
Regras de Negócio	
Fato Estrutural:	Informações relativas ao Servidor Web, ou IP da máquina remota que será acessada pelo aplicativo, e informações sobre o Servidor OPC que a máquina remota está utilizando deverão ser armazenadas.
Ação Restritiva:	Caso alguma informação tenha sido utilizada em outra parte do aplicativo, ela não poderá ser excluída.
Gatilho de Ação:	
Interface:	A interface deverá listar as informações cadastradas anteriormente pelo Usuário Cliente.
Cálculo:	

3.3.11 CASO DE USO: MANTER DISPOSITIVO

Descrição:	Este caso de uso começa quando o Usuário Cliente inicia o aplicativo que manipula as informações dos dispositivos de campo. O objetivo deste caso de uso é efetuar as operações relacionadas à inserção, alteração, pesquisa e exclusão de informações de dispositivos de campo. Ele termina quando o Usuário Cliente encerra o aplicativo dos dispositivos de campo.
Atores:	Usuário Cliente.
Pré-condições:	Ao menos um local remoto deverá estar cadastrado no sistema para ser utilizado como local onde se encontra o dispositivo de campo.
Caminho Primário	
Descrição:	**Inserir dispositivo de campo.**
Detalhe:	1 - Ator seleciona local cadastrado anteriormente onde está o dispositivo de campo. 2 - Ator informa dados sobre dispositivo de campo e efetua a inserção. 3 - Sistema verifica se informações já foram cadastradas anteriormente. 4 - Sistema armazena as informações. 5 - Sistema lista as informações de dispositivos já cadastrados no sistema.
Caminhos Alternativos	
Descrição:	**Alterar dispositivo de campo.**
Detalhe:	1 - Ator seleciona um dispositivo cadastrado anteriormente. 2 - Ator altera as informações e efetua a atualização. 3 - Sistema verifica se informações já foram cadastradas anteriormente. 4 - Sistema atualiza as informações. 5 - Sistema lista as informações de dispositivos já cadastrados no sistema.
Descrição:	**Pesquisar dispositivo de campo.**
Detalhe:	1 - Ao abrir o aplicativo, Ator deverá receber a listagem dos dispositivos de campo já cadastrados no sistema.
Caminhos de Exceção	
Descrição:	**Excluir dispositivo de campo.**
Detalhe:	1 - Ator seleciona um dispositivo cadastrado anteriormente. 2 - Ator efetua a exclusão. 3 - Sistema verifica se as informações já foram utilizadas anteriormente no sistema. 4 - Sistema exclui as informações. 5 - Sistema lista as informações de dispositivos já cadastrados no sistema.
Regras de Negócio	
Fato Estrutural:	Deverá armazenar o local onde o dispositivo está instalado, ou onde possa ser acessado, como também a tag registrada para o dispositivo.
Ação Restritiva:	Caso alguma informação tenha sido utilizada em outra parte do aplicativo, ela não poderá ser excluída.
Gatilho de Ação:	
Interface:	A interface deverá listar as informações já cadastradas anteriormente pelo Usuário Cliente.
Cálculo:	

3.3.12 CASO DE USO: MANTER TAG

Descrição:	Este caso de uso começa quando o Usuário Cliente inicia o aplicativo que manipula as informações de tags de dispositivos de campo. O objetivo deste caso de uso é efetuar as operações relacionadas à inserção, alteração, pesquisa e exclusão de informações relativas às tags de dispositivos de campo. Ele termina quando o Usuário Cliente encerra o aplicativo de tags.
Atores:	Usuário Cliente.
Pré-condições:	Pelo menos um dispositivo de campo deverá estar cadastrado no sistema para ser utilizado como o dispositivo das tags cadastradas.
Caminho Primário	
Descrição:	**Inserir tags de dispositivos de campo.**
Detalhe:	1 - Ator seleciona o dispositivo de campo cadastrado anteriormente que será usado como dispositivo das tags. 2 - Ator informa os dados sobre as tags e efetua a inserção. 2 - Sistema verifica se as informações já foram cadastradas anteriormente. 3 - Sistema armazena as informações. 4 - Sistema lista as informações de tags já cadastradas no sistema.
Caminhos Alternativos	
Descrição:	**Alterar tags de dispositivo de campo.**
Detalhe:	1 - Ator seleciona tag cadastrada anteriormente. 2 - Ator altera as informações e efetua a atualização. 3 - Sistema verifica se informações já foram cadastradas anteriormente. 4 - Sistema atualiza as informações. 5 - Sistema lista as informações de tags já cadastradas no sistema.
Descrição:	**Pesquisar tags de dispositivo de campo.**
Detalhe:	1 - Ao abrir o aplicativo, Ator deverá receber a listagem de tags de dispositivos de campo já cadastradas no sistema.
Caminhos de Exceção	
Descrição:	**Excluir tags de dispositivo de campo.**
Detalhe:	1 - Ator seleciona tag cadastrada anteriormente. 2 - Ator efetua a exclusão. 3 - Sistema verifica se informações já foram utilizadas anteriormente no sistema. 4 - Sistema exclui as informações. 5 - Sistema lista as informações de dispositivos já cadastrados no sistema.
Regras de Negócio	
Fato Estrutural:	
Ação Restritiva:	Caso alguma informação tenha sido utilizada em outra parte do aplicativo, ela não poderá ser excluída.
Gatilho de Ação:	
Interface:	A interface deverá listar as informações já cadastradas anteriormente pelo Usuário Cliente.
Cálculo:	

3.3.13 CASO DE USO: SELECIONAR MONITORAMENTO

Descrição:	Este caso de uso começa quando o Usuário Cliente inicia o aplicativo de seleção de monitoramento. O objetivo deste caso de uso é permitir ao Usuário Cliente filtrar e armazenar as informações necessárias ao monitoramento remoto. Ele termina quando o Usuário Cliente encerra o aplicativo de seleção de monitoramento.
Atores:	Usuário Cliente.
Pré-condições:	
Caminho Primário	
Descrição:	**Selecionar parâmetros de monitoramento remoto.**
Detalhe:	1 - Ator seleciona informações. 1.1 - Ator seleciona local que será monitorado. 1.2 - Ator seleciona um dos dispositivos cadastrados para local escolhido no item 1.1. 1.3 - Ator seleciona tags do dispositivo escolhido no item 1.2. 1.4 - Ator insere dados selecionados. 2 - Ator repete passo 1 até satisfazer as necessidades de monitoramento. 3 - Ator insere dados selecionados.
Caminhos Alternativos	
Descrição:	
Detalhe:	
Caminhos de Exceção	
Descrição:	**Excluir parâmetros de monitoramento remoto.**
Detalhe:	1 - Ator escolhe uma seleção cadastrada anteriormente. 2 - Ator efetua a exclusão. 3 - Sistema exclui as informações. 4 - Sistema lista informações de locais remotos já cadastrados no sistema.
Regras de Negócio	
Fato Estrutural:	Todas as informações necessárias para o monitoramento remoto deverão ser copiadas de suas tabelas de origem para uma única tabela para permitir um melhor ganho de desempenho.
Ação Restritiva:	Caso alguma informação tenha sido utilizada em outra parte do aplicativo, ela não poderá ser excluída.
Gatilho de Ação:	
Interface:	A interface deverá listar as informações já cadastradas anteriormente pelo Usuário Cliente.
Cálculo:	

3.3.14 CASO DE USO: INICIAR MONITORAMENTO

Descrição:	Este caso de uso começa quando o Usuário Cliente inicia o aplicativo que efetua o monitoramento remoto de tags. O objetivo deste caso de uso está relacionado ao envio, recebimento, processamento e retorno de informações entre o Módulo Cliente e o Módulo Servidor. Este caso de uso termina ao receber a resposta do Módulo Servidor referente ao recebimento das tags a serem monitoradas.
Atores:	Usuário Cliente, Módulo Cliente, Módulo Servidor, Temporizador Monitoramento e Temporizador Ping, Servidor OPC.
Pré-condições:	Os Módulos Cliente e Servidor estão em pleno funcionando.
Caminho Primário	
Descrição:	**Iniciar monitoramento remoto.**
Detalhe:	1 - Ator Usuário Cliente escolhe uma das seleções efetuadas anteriormente, descrita no caso de uso "Selecionar Monitoramento", para ser monitorada. 2 - Ator Usuário Cliente dispara a monitoramento. 3 - Módulo Cliente separa e agrupa as tags por IP. Cada IP indica para qual Módulo Servidor as informações serão enviadas. 4 - Módulo Cliente envia para cada Módulo Servidor as tags a serem monitoradas. 5 - Módulo Cliente inicia o Temporizador Monitoramento. 6 - Módulo Servidor recebe as tags e as adiciona no Servidor OPC para serem monitoradas. Módulo Servidor recebe, como retorno do Servidor OPC, o *GroupHandle* que representa o identificador do grupo de tags monitoradas. 7 - Módulo Servidor inicia o Temporizador Ping. 8 - Módulo Servidor envia *GroupHandle* para Módulo Cliente. 9 - Módulo Cliente recebe *GroupHandle* e armazena nas respectivas tags.
Caminhos Alternativos	
Descrição:	
Detalhe:	
Caminhos de Exceção	
Descrição:	**Falha no envio de tags para o Módulo Servidor.**
Detalhe:	1 - Efetuar os passos de 1 a 4 do caminho primário. 2 - Módulo Cliente tenta enviar informações para Módulo Servidor, mas não consegue fazê-lo dentro do tempo pré-estabelecido (*Timeout*). 3 - Sistema informa que não foi possível enviar tags entre Módulo Cliente e Módulo Servidor.
Regras de Negócio	
Fato Estrutural:	O tempo pré-estabelecido para envio dos dados (*Timeout*) deve ser 20s.
Ação Restritiva:	
Gatilho de Ação:	
Interface:	Tags utilizadas no monitoramento devem aparecer na tela em forma de lista. Tags devem ser mostradas na cor vermelha na lista até receber seu respectivo *GroupHandle*. Ao receber o *GroupHandle*, as tags passam a ser mostradas na cor preta.
Cálculo:	

3.3.15 CASO DE USO: ATUALIZAR DADOS MONITORADOS

Descrição:	Este caso de uso começa quando o Servidor OPC recebe tags para serem monitoradas e entra em um ciclo infinito. A forma como este processo termina está descrita no caso de uso "Cancelar Monitoramento".
Atores:	Servidor OPC, Módulo Servidor e Módulo Cliente.
Pré-condições:	As tags devem ter sido adicionadas no Servidor OPC para serem monitoradas.
Caminho Primário	
Descrição:	**Atualizar dados monitorados.**
Detalhe:	1 - Servidor OPC envia o evento *DataChange* para Módulo Servidor, informando que algum valor de tag foi alterado. 2 - Módulo Servidor envia os valores recebidos para Módulo Cliente. 3 - Módulo Cliente recebe e armazena os valores nas respectivas tags.
Caminhos Alternativos	
Descrição:	
Detalhe:	
Caminhos de Exceção	
Descrição:	
Detalhe:	
Regras de Negócio	
Fato Estrutural:	
Ação Restritiva:	
Gatilho de Ação:	
Interface:	
Cálculo:	

3.3.16 CASO DE USO: PESQUISAR DADOS MONITORADOS

Descrição:	Este caso de uso começa quando o "Temporizador Monitor" é inicializado e entra em um ciclo infinito. A forma como este processo termina está descrita no caso de uso "Cancelar Monitoramento".
Atores:	Temporizador Monitoramento.
Pré-condições:	As tags devem estar sendo monitoradas.
Caminho Primário	
Descrição:	**Obter e exibir dados monitorados.**
Detalhe:	1 - Ator Temporizador Monitor obtém dados armazenados, que estão sendo constantemente atualizados pelo caso de uso "Atualizar Dados Monitorados". 2 - Ator Temporizador Monitor exibe dados obtidos.
Caminhos Alternativos	
Descrição:	
Detalhe:	
Caminhos de Exceção	
Descrição:	
Detalhe:	
Regras de Negócio	
Fato Estrutural:	Tempo entre pings deve ser 20s (ver caso de uso “Checar Comunicação”).
Ação Restritiva:	
Gatilho de Ação:	
Interface:	Opção para definir o intervalo de tempo utilizado pelo temporizador para obter e exibir os dados na tela deverá existir. Se valor de *VlrIntervalo* (ver item Cálculo) for superior a 20s, então as informações da tela referentes ao servidor remoto em questão deverão ser mostradas na cor vermelha, indicando que a comunicação caiu.
Cálculo:	O temporizador deverá calcular o intervalo de tempo entre a data atual e a última data de ping armazenada, para verificar se a comunicação está funcionando corretamente. *VlrIntervalo = DataAtual - UltimoPing* Onde: *VlrIntervalo*, *DataAtual* e *UltimoPing* são dados em segundos.

3.3.17 CASO DE USO: CHECAR COMUNICAÇÃO

Descrição:	Este caso de uso começa quando o "Temporizador Ping" é inicializado e entra em um ciclo infinito. A forma como este processo termina está descrita no caso de uso "Cancelar Monitoramento".
Atores:	Temporizador Ping, Módulo Cliente.
Pré-condições:	As tags devem estar sendo monitoradas.
Caminho Primário	
Descrição:	**Enviar sinal de Ping.**
Detalhe:	1 - Ator Temporizador Ping envia sinal de ping para Módulo Cliente. 2 - Módulo Cliente recebe o sinal de Ping e armazena as respectivas tags.
Caminhos Alternativos	
Descrição:	
Detalhe:	
Caminhos de Exceção	
Descrição:	
Detalhe:	
Regras de Negócio	
Fato Estrutural:	
Ação Restritiva:	
Gatilho de Ação:	
Interface:	
Cálculo:	

3.3.18 CASO DE USO: CANCELAR MONITORAMENTO

Descrição:	Este caso de uso começa quando o Usuário Cliente efetua o cancelamento do monitoramento remoto de tags. O objetivo deste caso de uso está relacionado ao envio de um comando entre o Módulo Cliente e o Módulo Servidor para cancelar o monitoramento remoto.
Atores:	Usuário Cliente, Módulo Cliente, Temporizador Monitoramento, Módulo Servidor, Servidor OPC e Temporizador Ping.
Pré-condições:	As tags devem estar sendo monitoradas.
Caminho Primário	
Descrição:	**Cancelar Monitoramento.**
Detalhe:	1 - Ator Usuário Cliente efetua o cancelamento do monitoramento. 2 - Módulo Cliente envia comando de cancelamento de monitoramento para Módulo Servidor. 3 - Módulo Cliente pára o Temporizador Monitoramento. 4 - Módulo Servidor recebe comando de cancelamento de monitoramento. 5 - Módulo Servidor informa Servidor OPC que o mesmo deve parar de monitorar tags. 6 - Módulo Servidor pára o Temporizador Ping.
Caminhos Alternativos	
Descrição:	
Detalhe:	
Caminhos de Exceção	
Descrição:	
Detalhe:	
Regras de Negócio	
Fato Estrutural:	
Ação Restritiva:	
Gatilho de Ação:	
Interface:	
Cálculo:	

3.4 DIAGRAMAS DE CLASSES

Considerando-se os casos de uso, as classes necessárias para o desenvolvimento do sistema foram identificadas e, para cada classe, foram relacionadas as operações necessárias para executar os procedimentos descritos nos casos de uso. A seção de atributos da classe não foi descrita pelo fato de que as informações, normalmente armazenadas nessa seção, serão armazenadas em banco de dados. A definição do sistema como sendo divido em três camadas, feita na arquitetura preliminar, também influenciou a identificação das classes.

As Figuras 42 e 43 mostram os diagramas de classes e os relacionamentos identificados para o desenvolvimento do software.

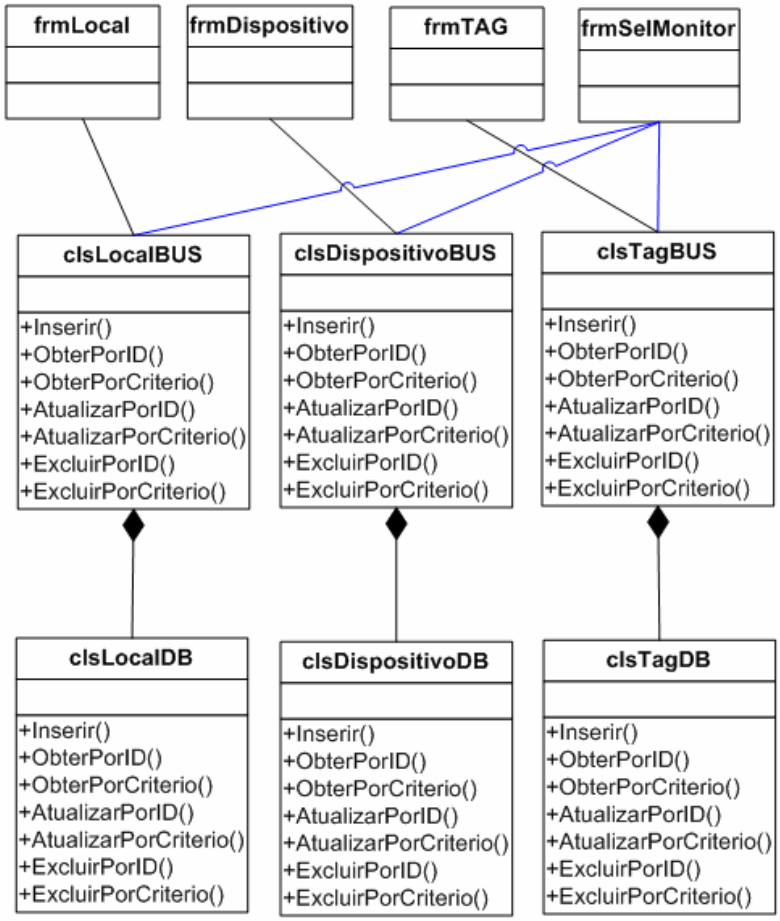

Figura 42 - Diagrama de classes - Parte "A"

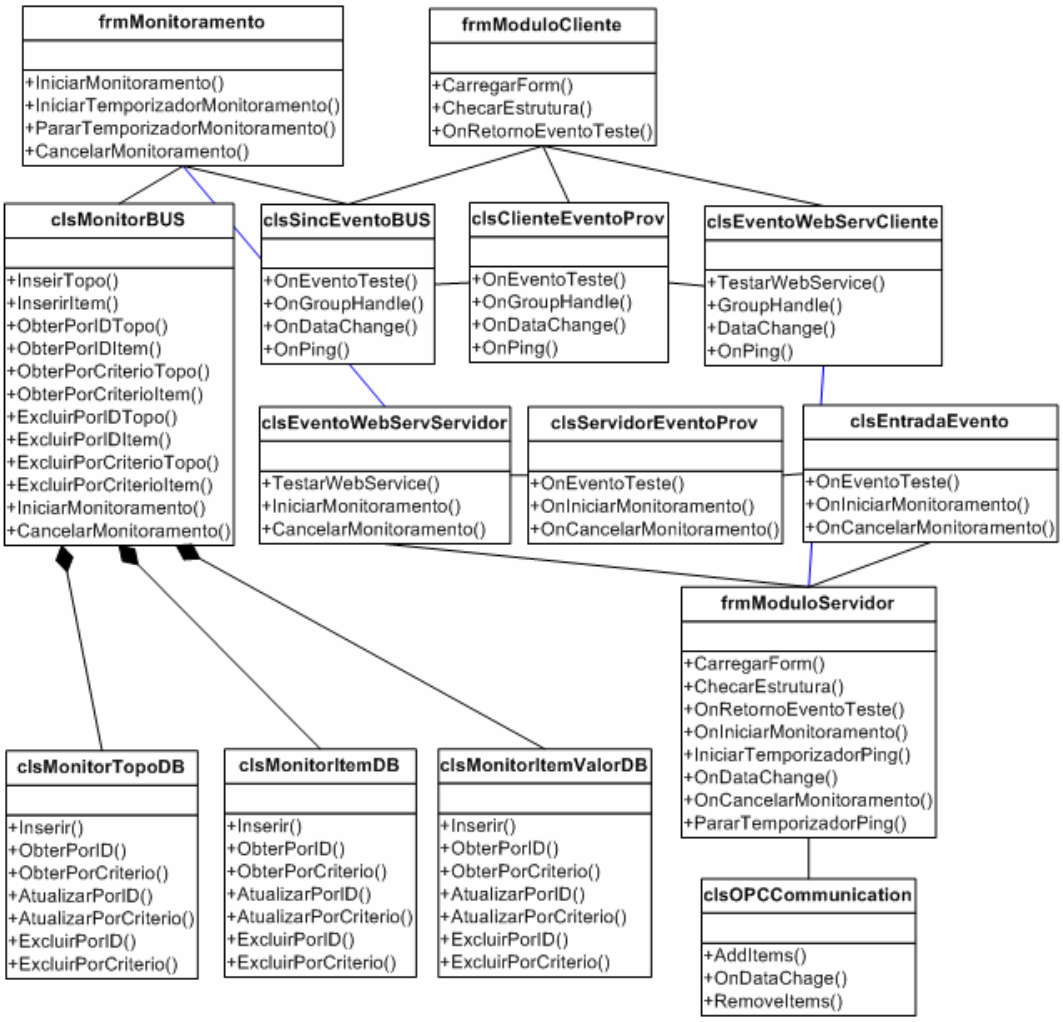

Figura 43 - Diagrama de classes - Parte "B"

3.5 DIAGRAMA DE COMPONENTES DO MÓDULO CLIENTE

O diagrama de componentes mostrado na Figura 44 ilustra o agrupamento das classes em componentes, e também a distribuição dos componentes dentro das camadas lógicas para o módulo cliente. O módulo cliente é representado pelo arquivo executável *EWS_ClienteAP.exe*. Através do aplicativo *EWS_ClienteAP.exe* serão executados os formulários: *frmModuloCliente*, *frmLocal*, etc. Estes formulários serão responsáveis pela manutenção das informações do sistemas e por efetuar o processo de monitoramento de dados descrito nos casos de uso.

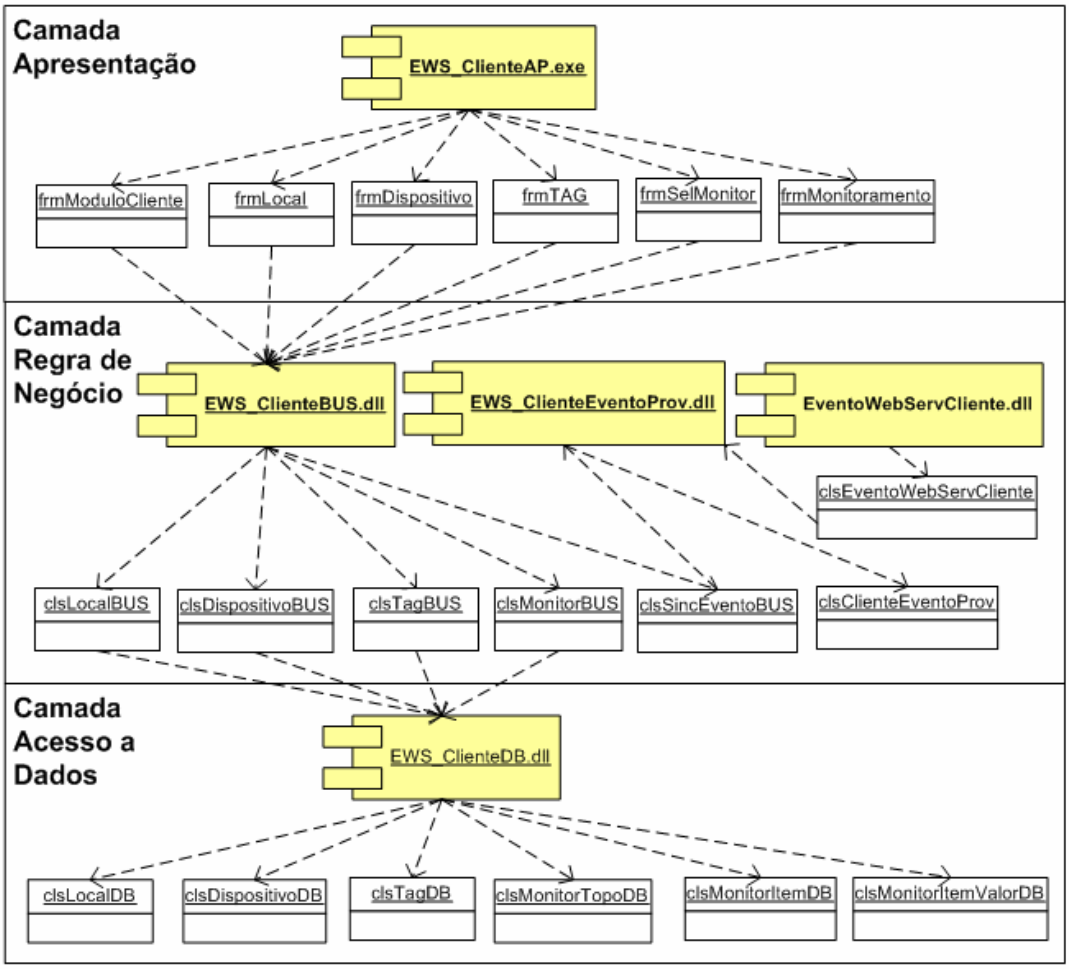

Figura 44 - Diagrama de componentes do módulo cliente

3.6 DIAGRAMA DE COMPONENTES DO MÓDULO SERVIDOR

A Figura 45 ilustra o diagrama de componentes para o módulo servidor. É importante notar que, no caso do módulo servidor, não será necessário utilizar nenhum armazenamento de dados, e por isso a "Camada de Acesso a Dados" não será implementada. O módulo servidor é representado pelo arquivo executável *EWS_ServidorAP.exe*. Este aplicativo possui apenas um formulário, *frmModuloServidor*, que será responsável pela troca de informações entre o módulo cliente e o módulo servidor.

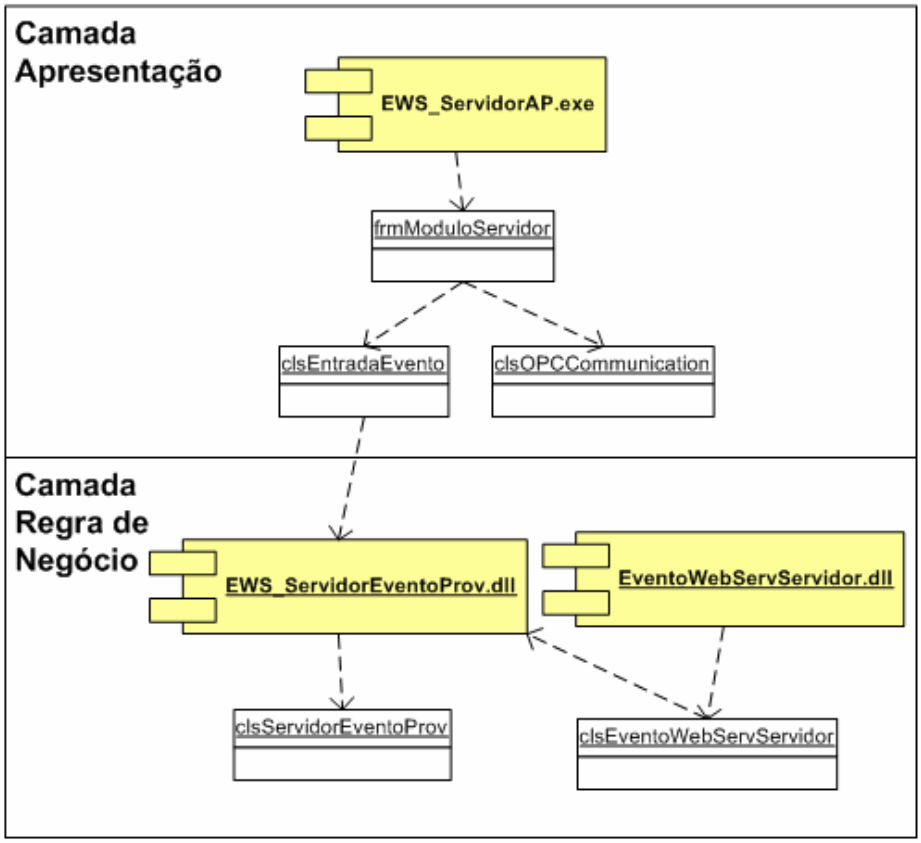

Figura 45 - Diagrama de componentes do módulo servidor

3.7 DIAGRAMAS DE SEQÜÊNCIA

Os diagramas de seqüência apresentam as interações entre os Atores (usuários do sistema) e as aplicações, conforme descritas nos casos de uso.

3.7.1 PARA O CASO DE USO "INICIAR MÓDULO CLIENTE"

A Figura 46 descreve as seqüências de ações ocorridas na inicialização do módulo cliente.

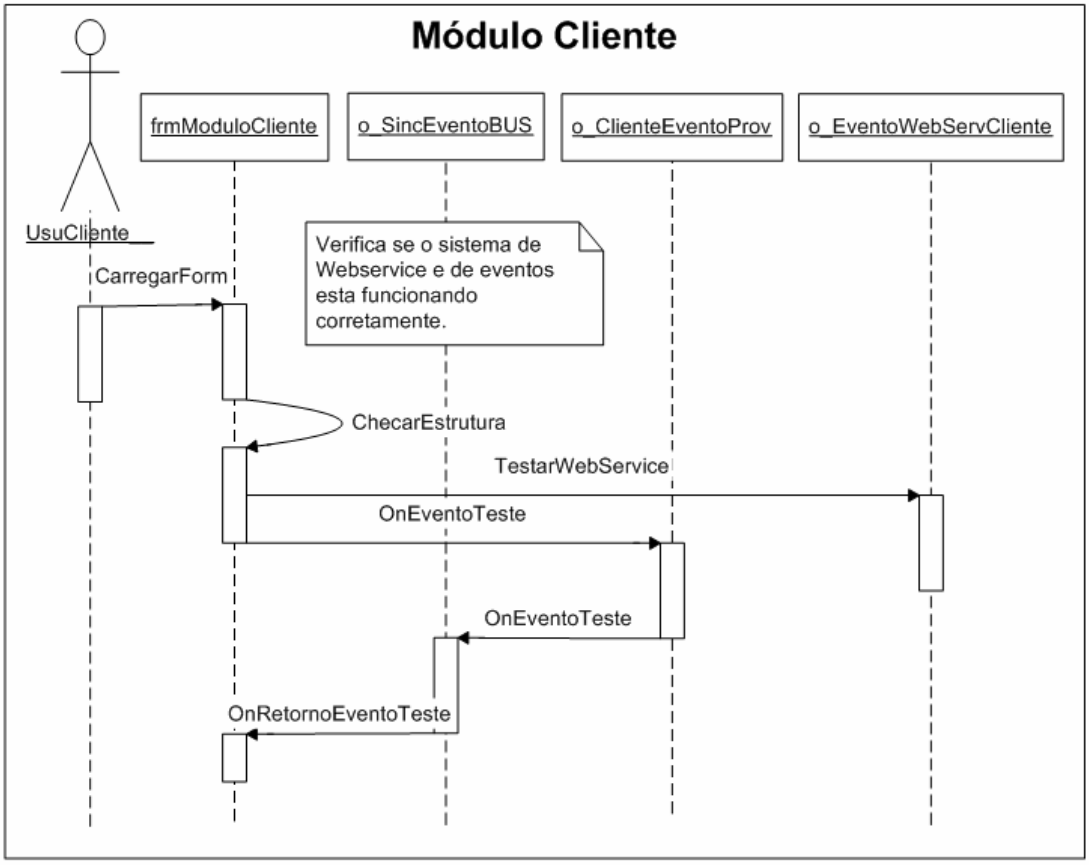

Figura 46 - Diagrama de seqüência "Iniciar Módulo Cliente"

3.7.2 PARA O CASO DE USO "INICIAR MÓDULO SERVIDOR"

A Figura 47 descreve as seqüências de ações ocorridas na inicialização do módulo servidor.

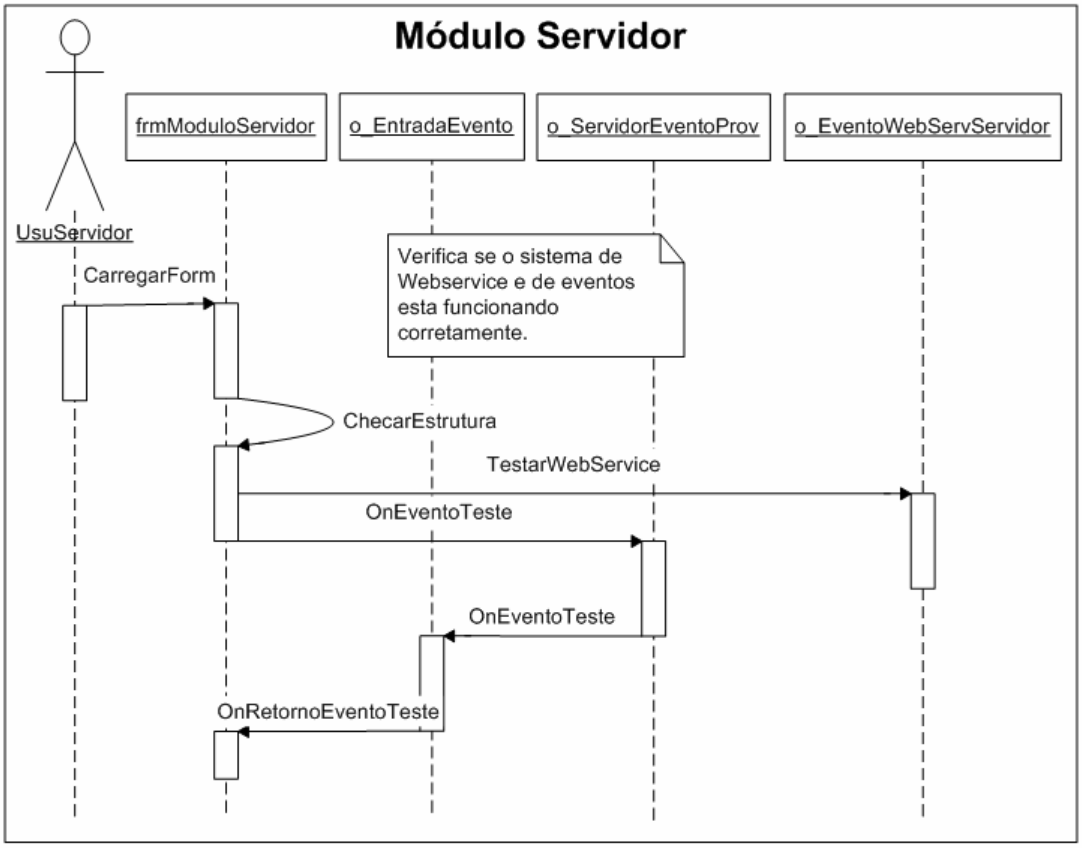

Figura 47 - Diagrama de seqüência "Iniciar Módulo Servidor"

3.7.3 PARA O CASO DE USO "MANTER LOCAL"

As Figuras 48 e 49 descrevem as seqüências de ações ocorridas na manutenção das informações do local onde se encontram os dispositivos de campo. O diagrama da Figura 48 descreve as ações "Inserir" e "Pesquisar" informações.

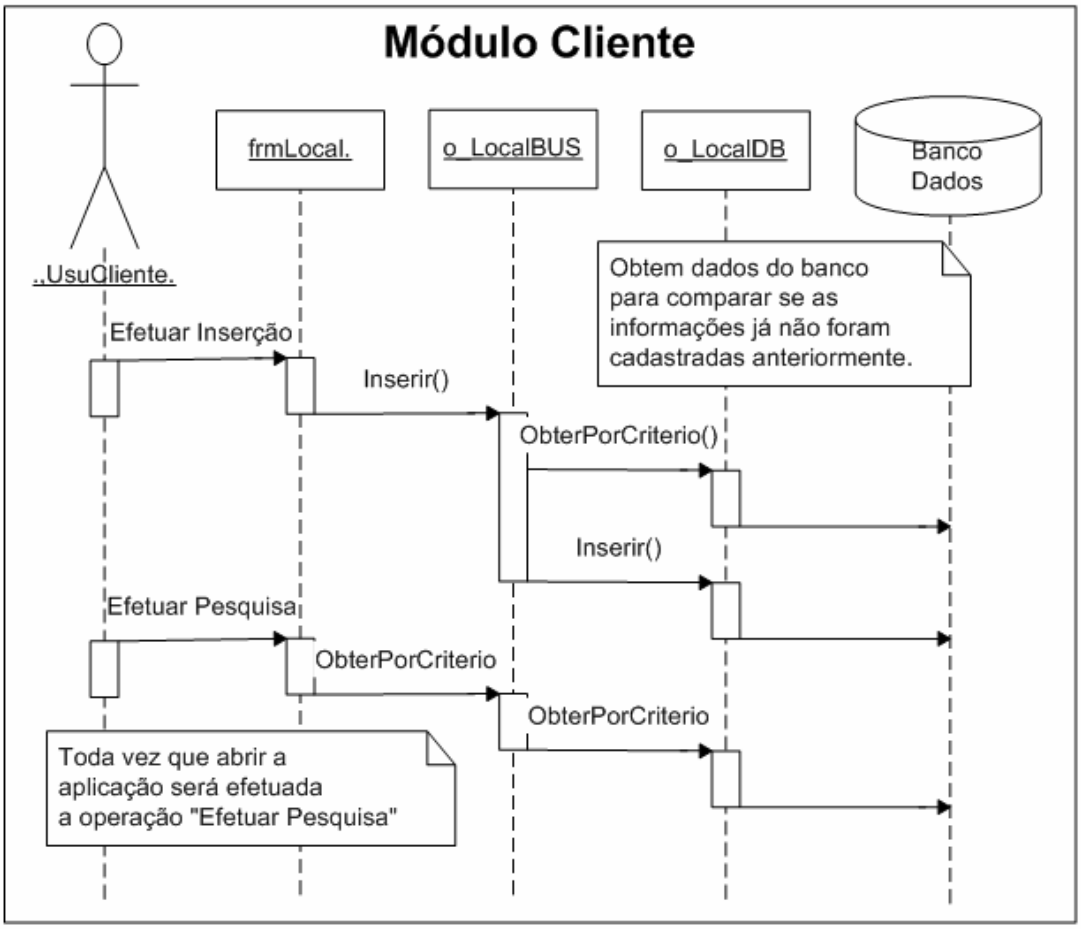

Figura 48 - Diagrama de seqüência "Manter Local" - ações: "Inserir" e "Pesquisar"

O diagrama mostrado na Figura 49 descreve as ações "Alterar" e "Excluir" informações para o caso de uso "Manter Local".

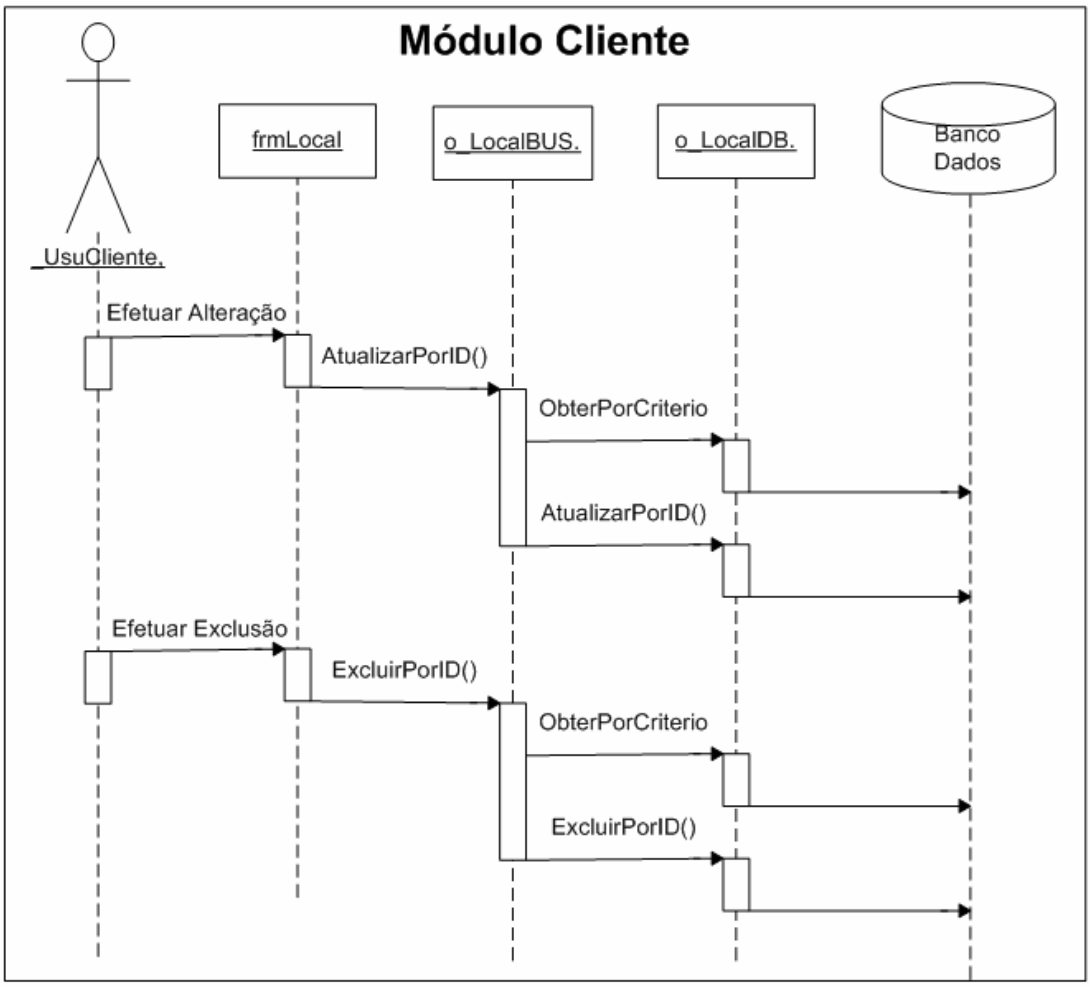

Figura 49 - Diagrama de seqüência "Manter Local" - ações: "Alterar" e "Excluir"

3.7.4 PARA O CASO DE USO "MANTER DISPOSITIVO"

As Figuras 50 e 51 descrevem as seqüências de ações ocorridas na manutenção de informações de dispositivos de campo. A Figura 50 descreve as ações "Inserir" e "Pesquisar" informações.

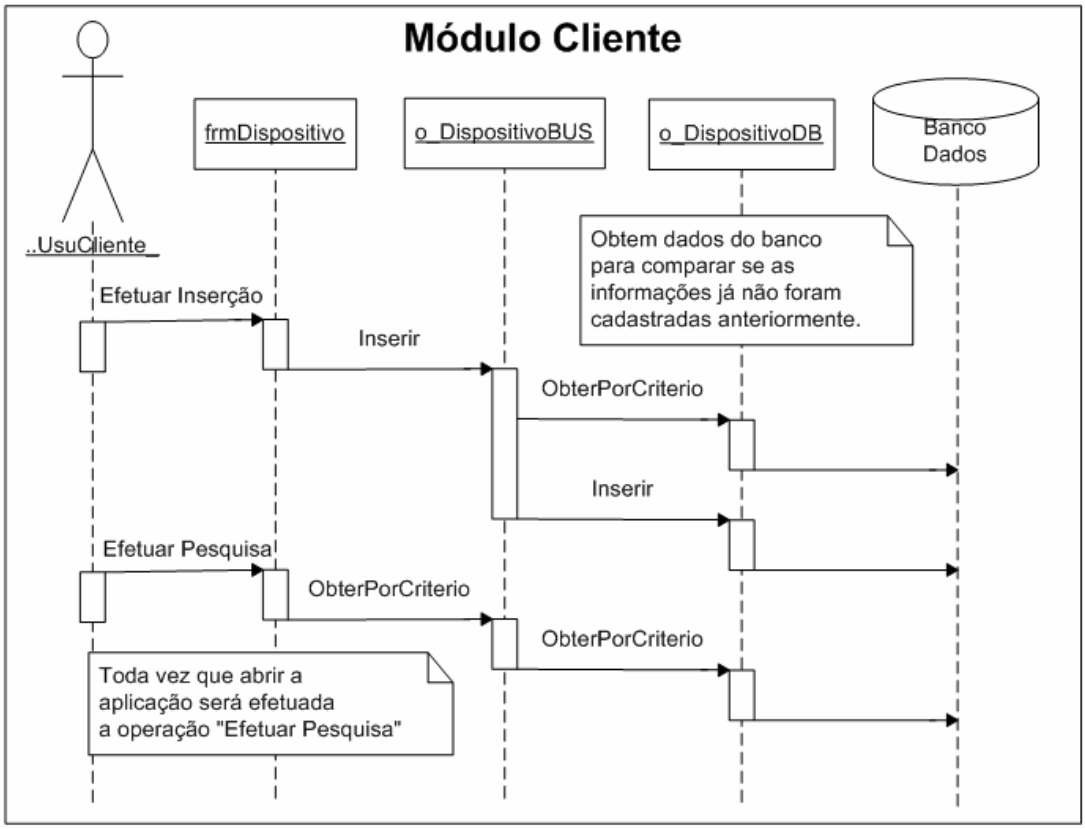

Figura 50 - Diagrama de seqüência "Manter Dispositivo" - ações: "Inserir" e "Pesquisar"

O diagrama mostrado na Figura 51 descreve as ações "Alterar" e "Excluir" informações para o caso de uso "Manter Dispositivo".

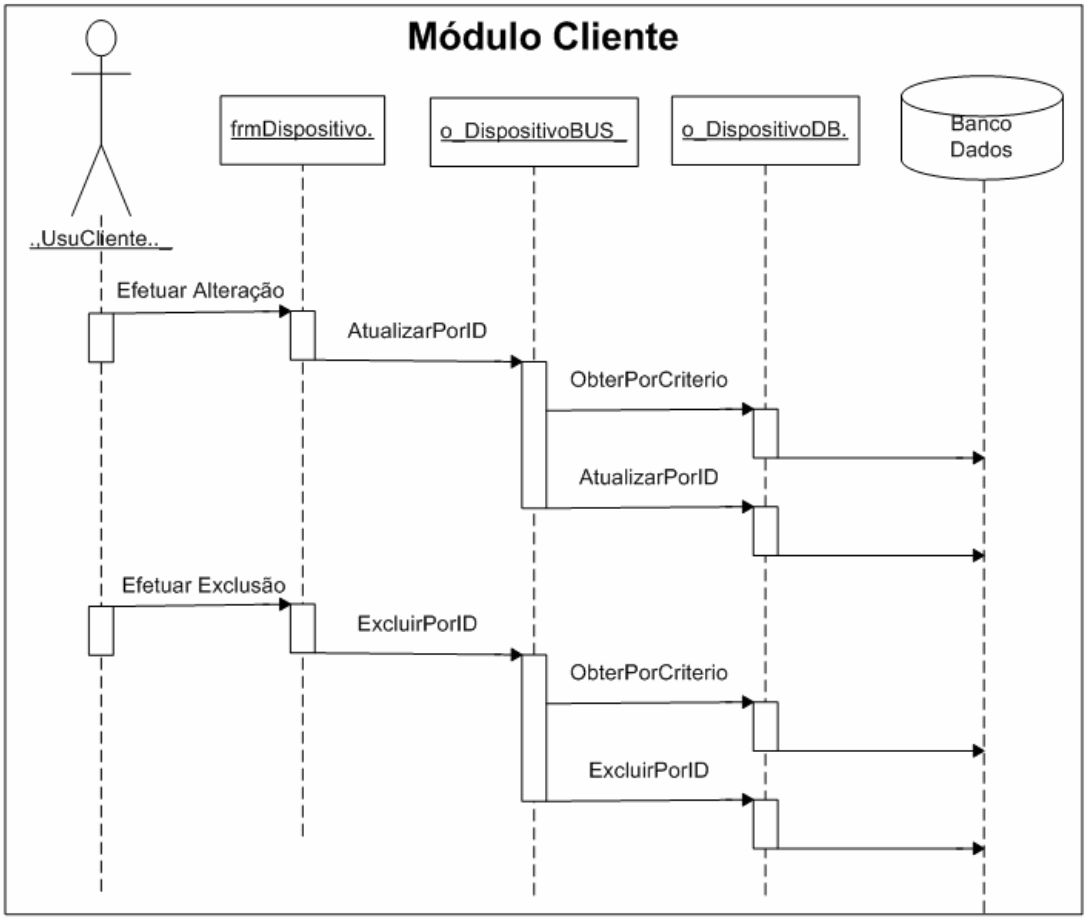

Figura 51 - Diagrama de seqüência "Manter Dispositivo" - ações: "Alterar" e "Excluir"

3.7.5 PARA O CASO DE USO "MANTER TAG"

As Figuras 52 e 53 descrevem as seqüências de ações ocorridas na manutenção de informações de tags de dispositivos de campo. A Figura 52 descreve as ações "Inserir" e "Pesquisar" informações.

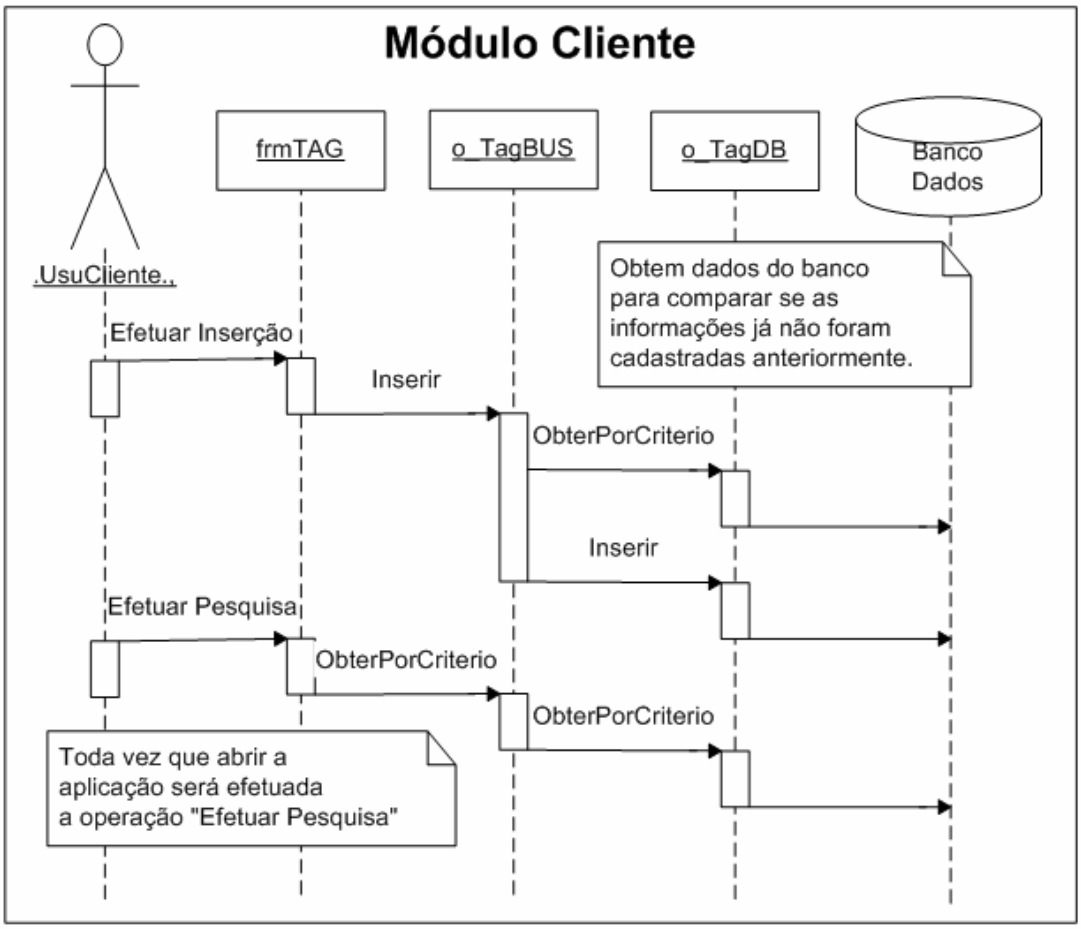

Figura 52 - Diagrama de seqüência "Manter Tag" - ações: "Inserir" e "Pesquisar"

O diagrama mostrado na Figura 53 descreve as ações "Alterar" e "Excluir" informações para o caso de uso "Manter Tag".

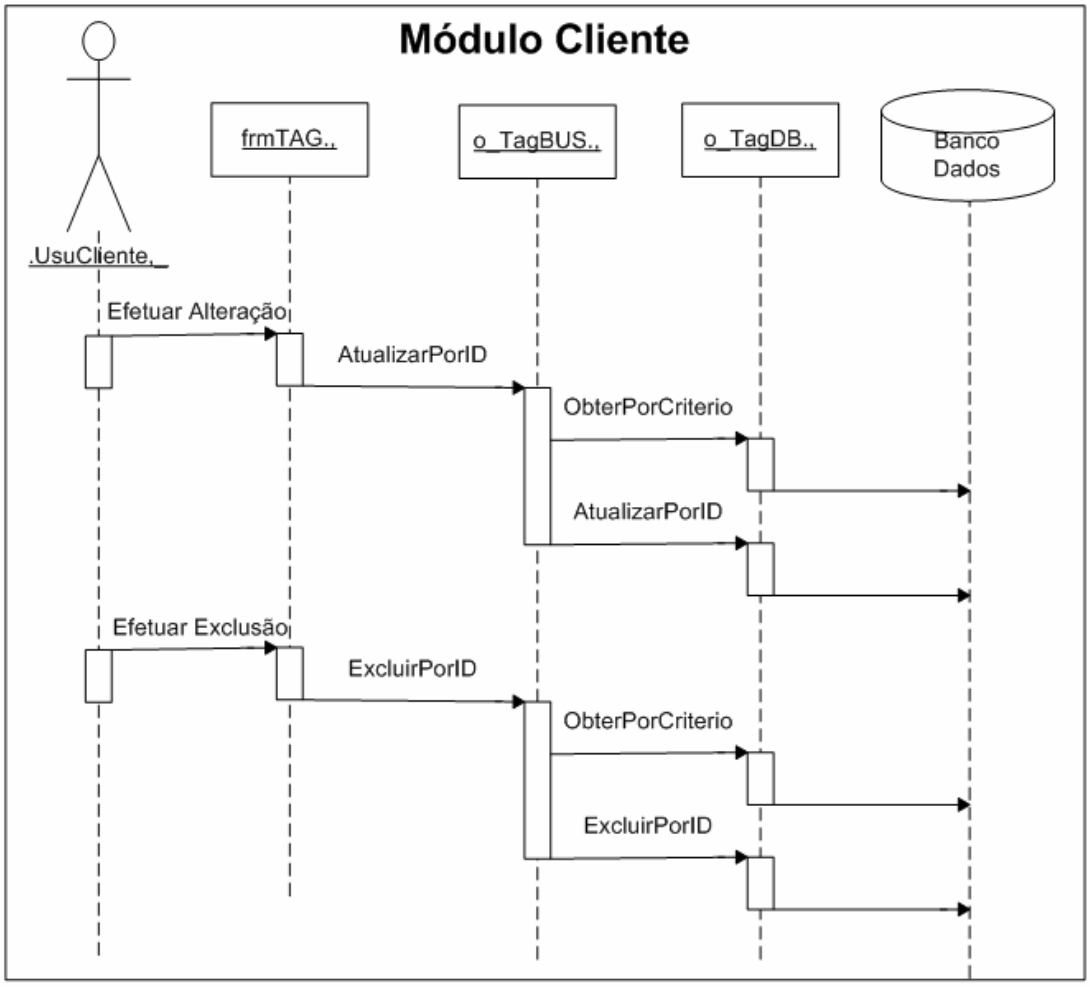

Figura 53 - Diagrama de seqüência "Manter Tag" - ações: "Alterar" e "Excluir"

3.7.6 PARA O CASO DE USO "SELECIONAR MONITORAMENTO"

As Figuras 54 e 55 descrevem as seqüências de ações que ocorrem na seleção de informações a serem monitoradas. Primeiramente, a Figura 54 descreve as ações para obter as informações do banco de dados, que foram cadastradas anteriormente, para que o usuário possa escolher as que devem ser monitoradas.

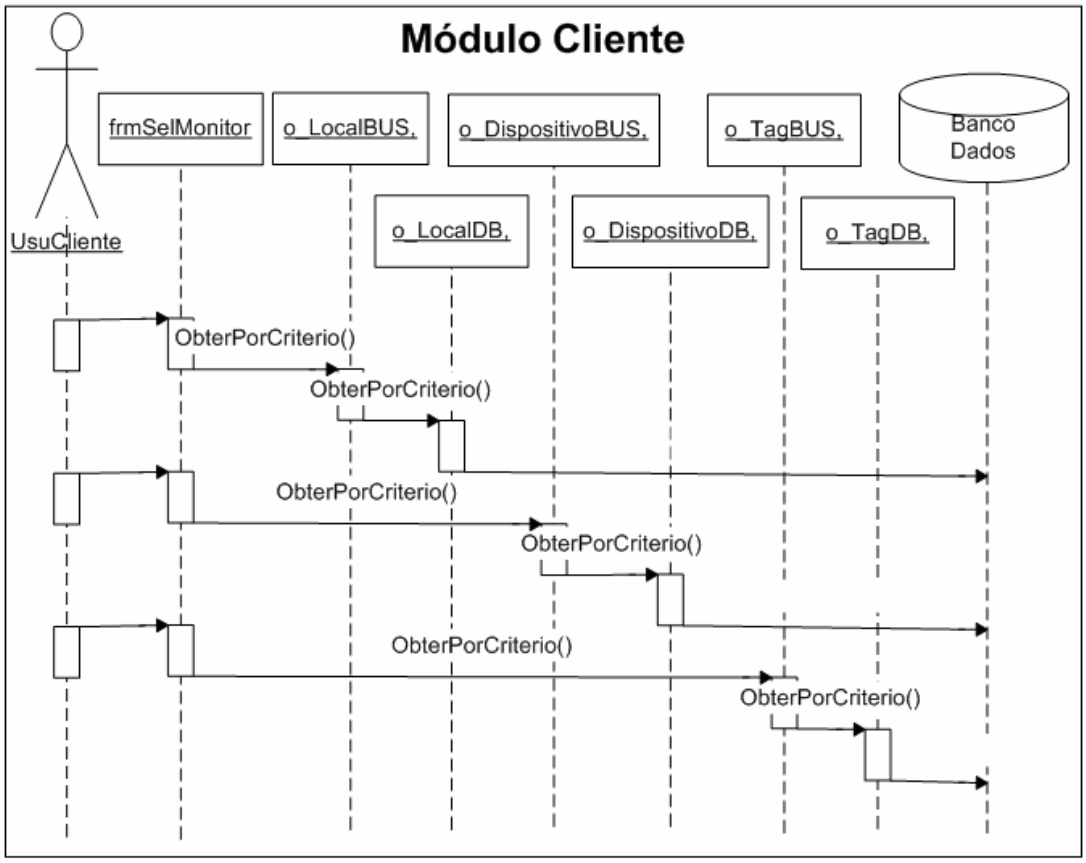

Figura 54 - Diagrama de seqüência "Selecionar Monitoramento" - ação: "Exibir dados para seleção"

Efetuadas as escolhas, o usuário salva as informações no banco de dados, como mostra a ação "Inserir" na Figura 55. A Figura 55 também mostra a pesquisa de informações de seleção já armazenadas no banco de dados, representada pela ação "Pesquisar".

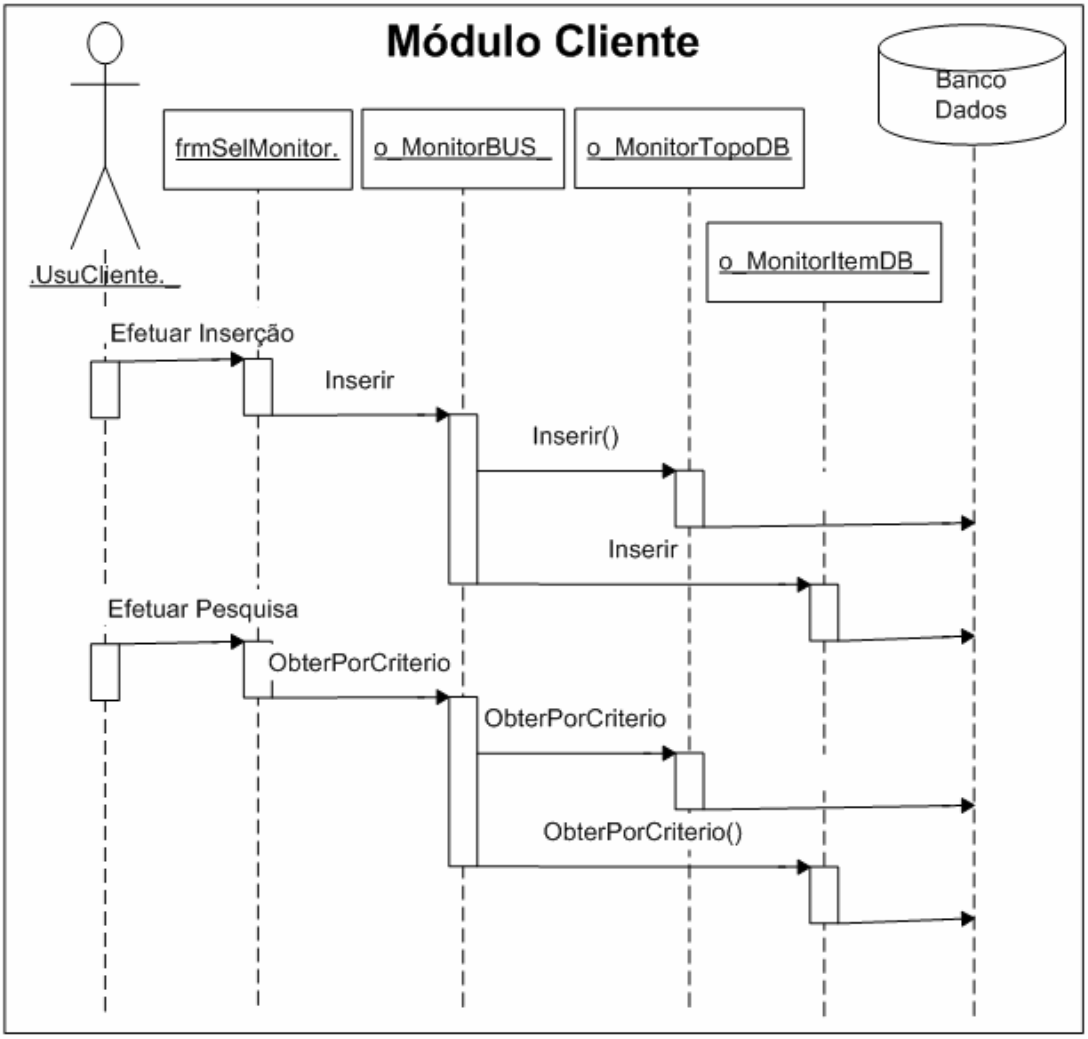

Figura 55 - Diagrama de seqüência "Selecionar Monitoramento" - ações: "Inserir" e "Pesquisar"

Por fim, a Figura 56 mostra a seqüência de ações para exclusão de informações cadastradas no banco de dados.

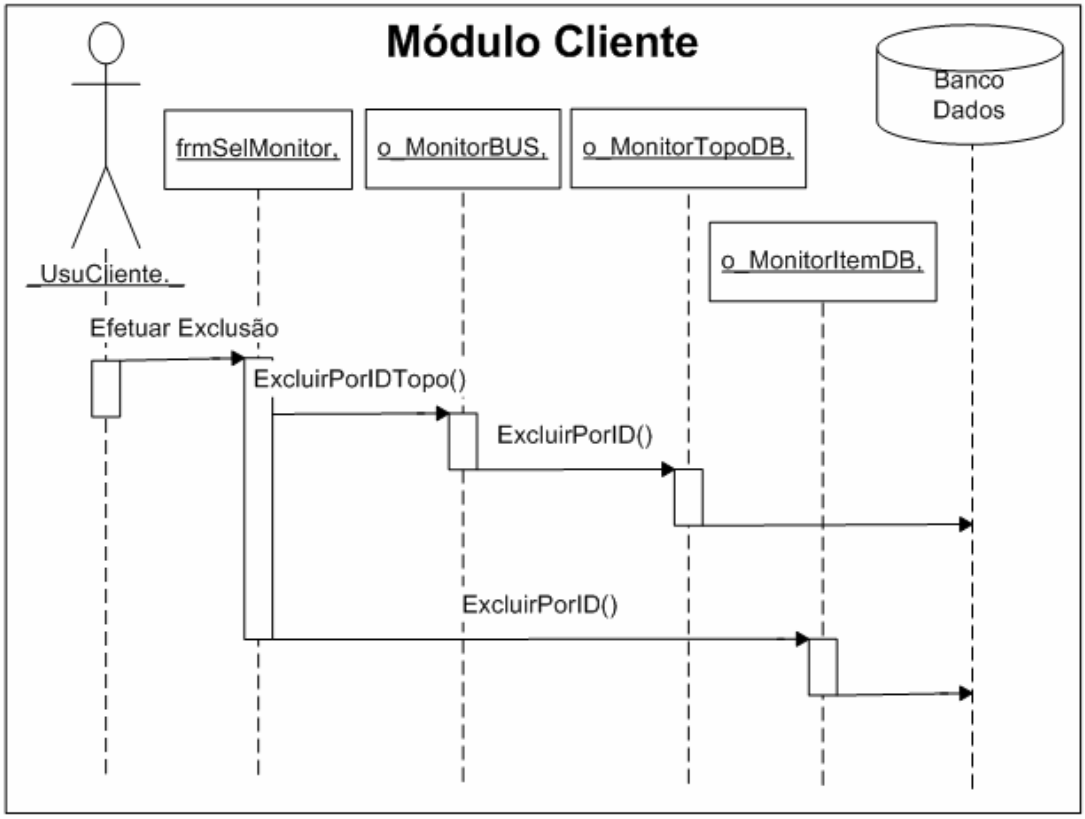

Figura 56 - Diagrama de seqüência "Selecionar Monitoramento" - ação: "Excluir"

3.7.7 PARA OS CASOS DE USO "INICIAR MONITORAMENTO" E "PESQUISAR DADOS MONITORADOS"

As Figuras 57 e 58 descrevem as seqüências de ações que ocorrem nos casos de uso "Iniciar o Monitoramento", para as tags, e "Pesquisar Dados Monitorados". O diagrama de seqüências foi dividido em duas partes, Parte I e Parte II, para facilitar seu entendimento.

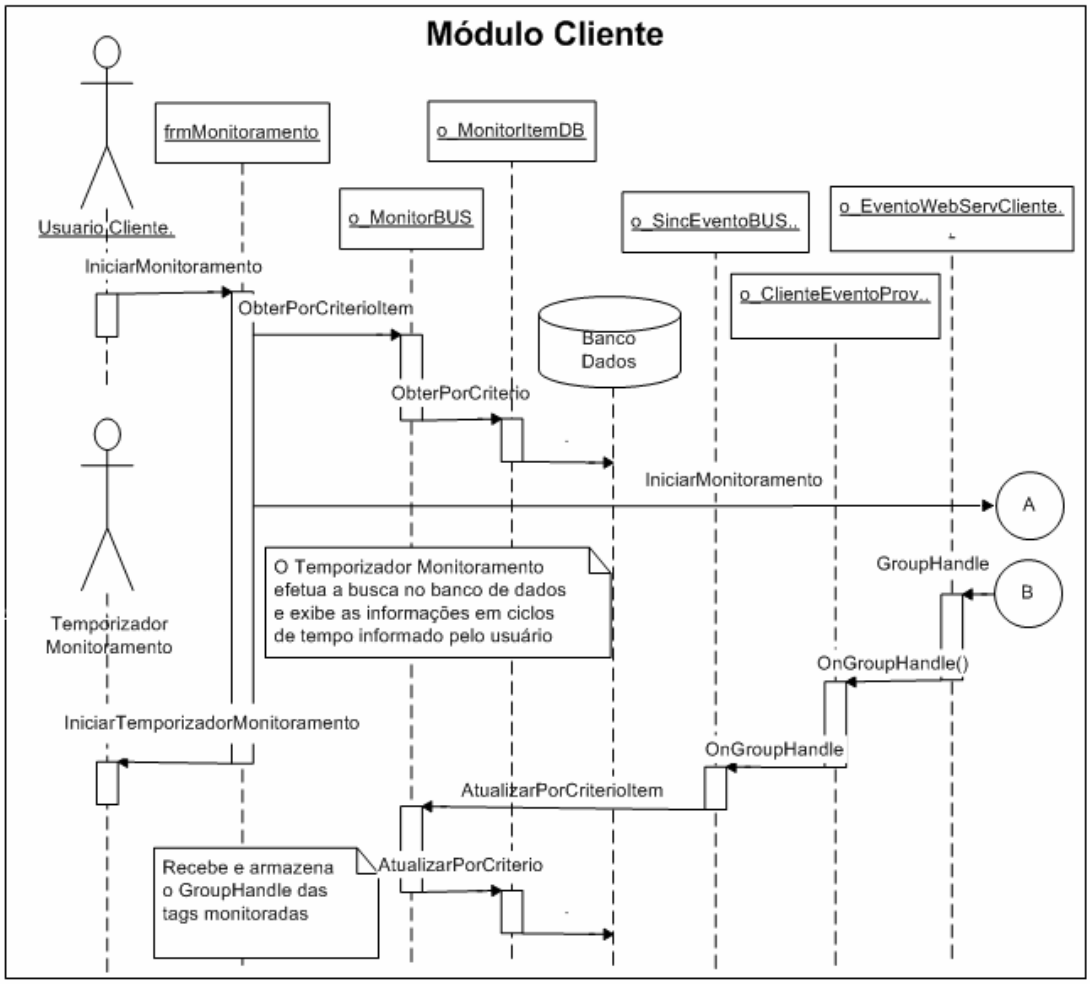

Figura 57 - Diagrama de seqüência "Iniciar Monitoramento" e "Pesquisar Dados Monitorados". Parte I

Os pontos de conexão entre as partes do diagrama são indicados, nas Figuras 57 e 58, pelas letras "A" e "B".

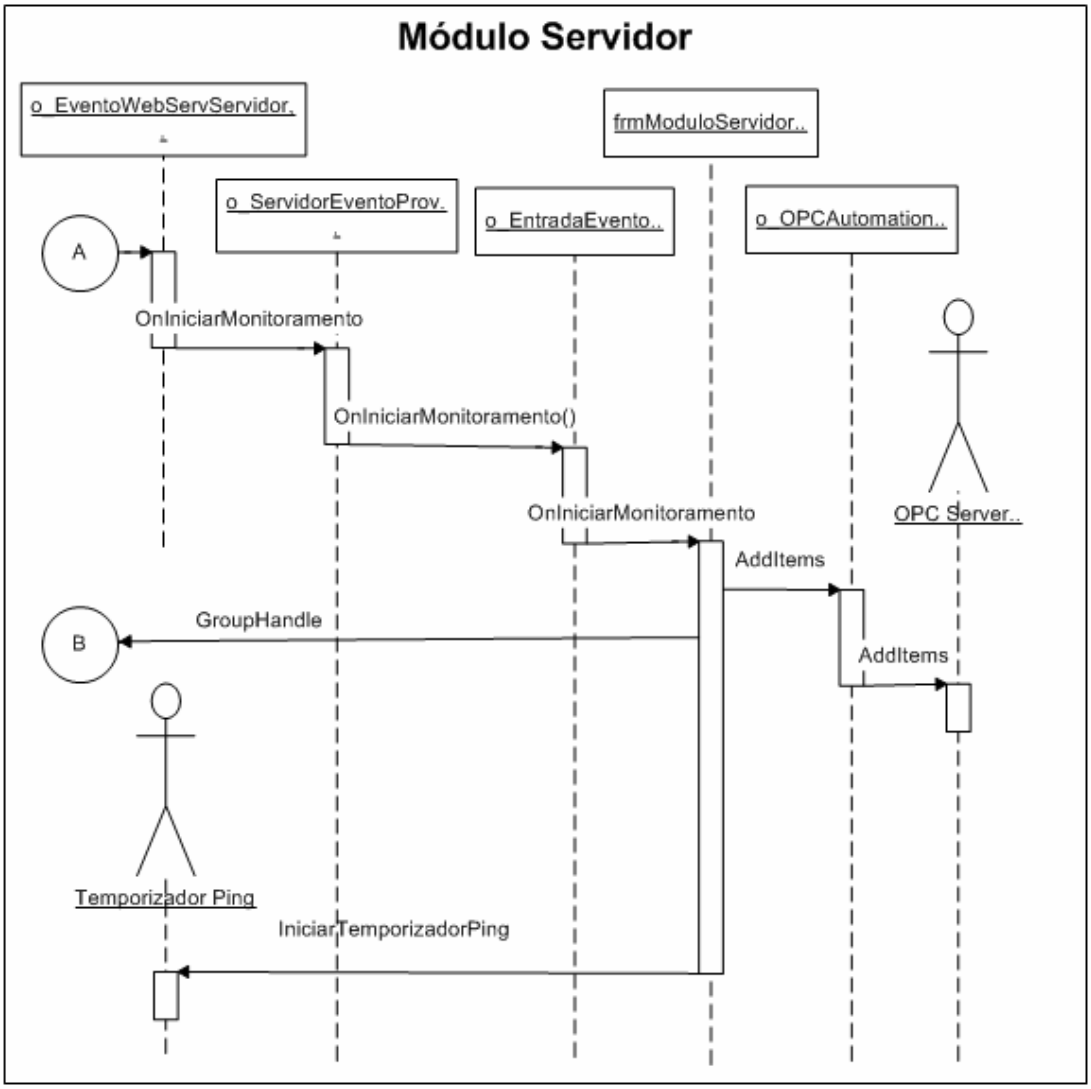

Figura 58 - Diagrama de seqüência "Iniciar Monitoramento" e "Pesquisar Dados Monitorados". Parte II

3.7.8 PARA OS CASOS DE USO "ATUALIZAR DADOS MONITORADOS" E "CHECAR COMUNICAÇÃO"

As Figuras 59 e 60 descrevem as seqüências de ações que ocorrem para os casos de uso "Atualizar Dados Monitorados" e "Checar Comunicação".

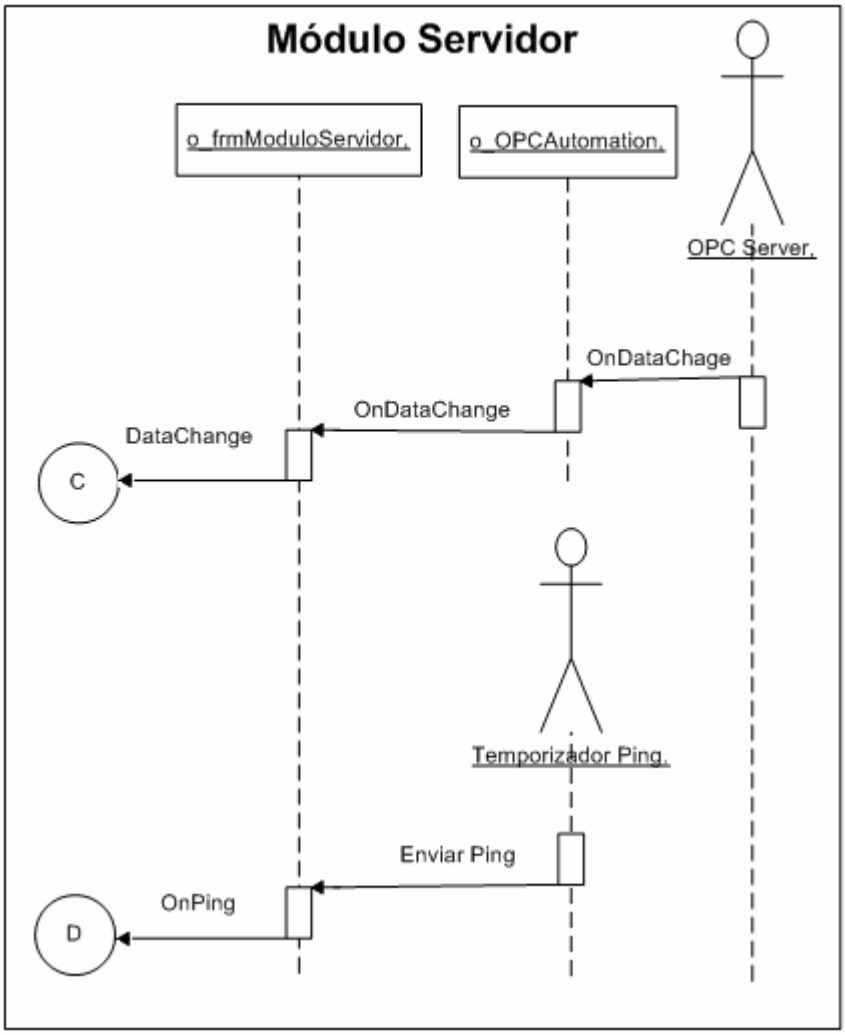

Figura 59 - Diagrama de seqüência "Atualizar Dados Monitorados" e "Checar Comunicação". Parte I

O diagrama de seqüências foi dividido em duas partes, Parte I e Parte II, para facilitar seu entendimento. Os pontos de conexão entre as partes do diagrama são indicados, nas Figuras 59 e 60, pelas letras "C" e "D".

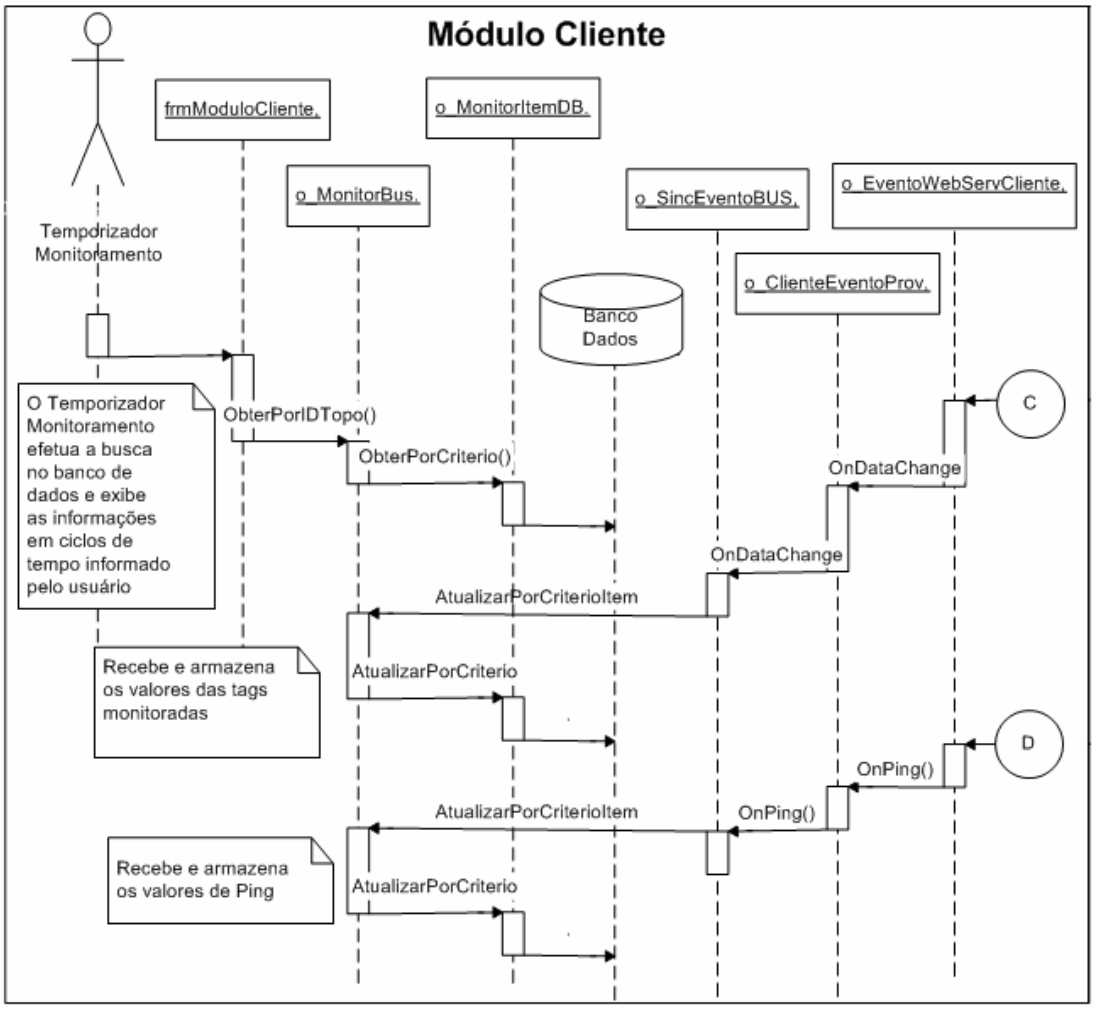

Figura 60 - Diagrama de seqüência "Atualizar Dados Monitorados" e "Checar Comunicação". Parte II

3.7.9 PARA O CASO DE USO "CANCELAR MONITORAMENTO"

A Figura 61 descreve as seqüências de ações que ocorrem para o caso de uso "Cancelar Monitoramento".

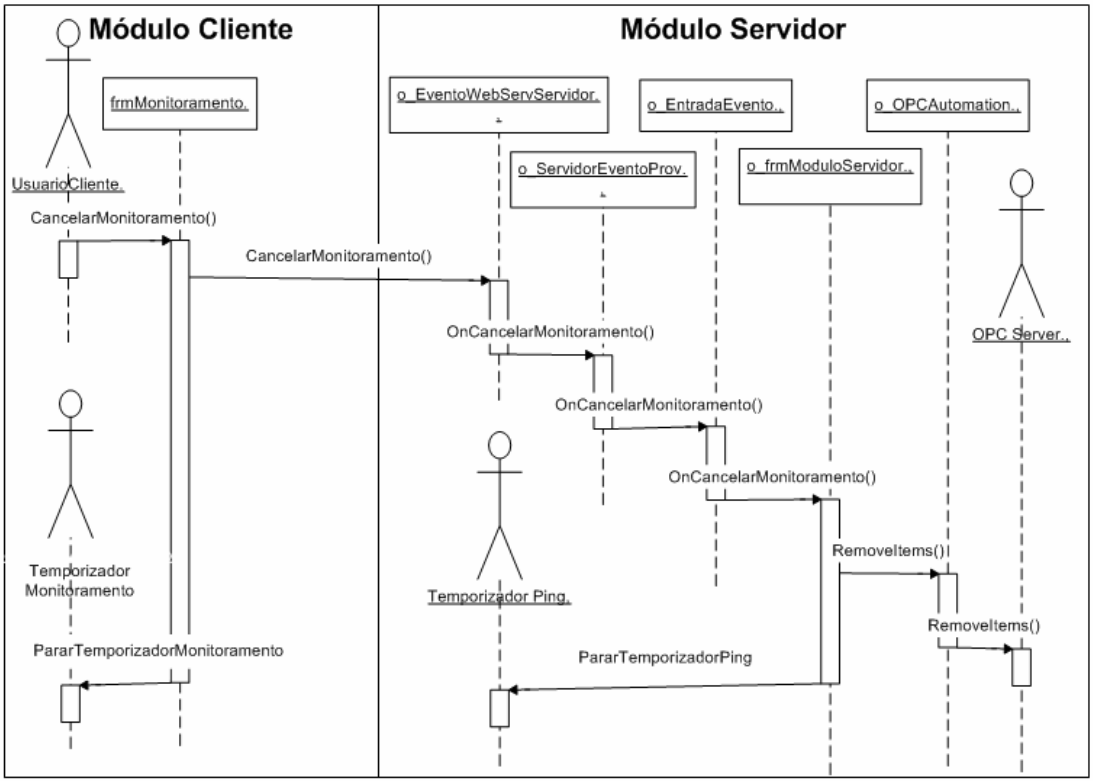

Figura 61 - Diagrama de seqüência "Cancelar Monitoramento"

3.8 DIAGRAMA DE IMPLANTAÇÃO

Os diagramas de implantação das Figuras 62 e 63 mostram a disposição do hardware e sua ligação com os módulos de software.

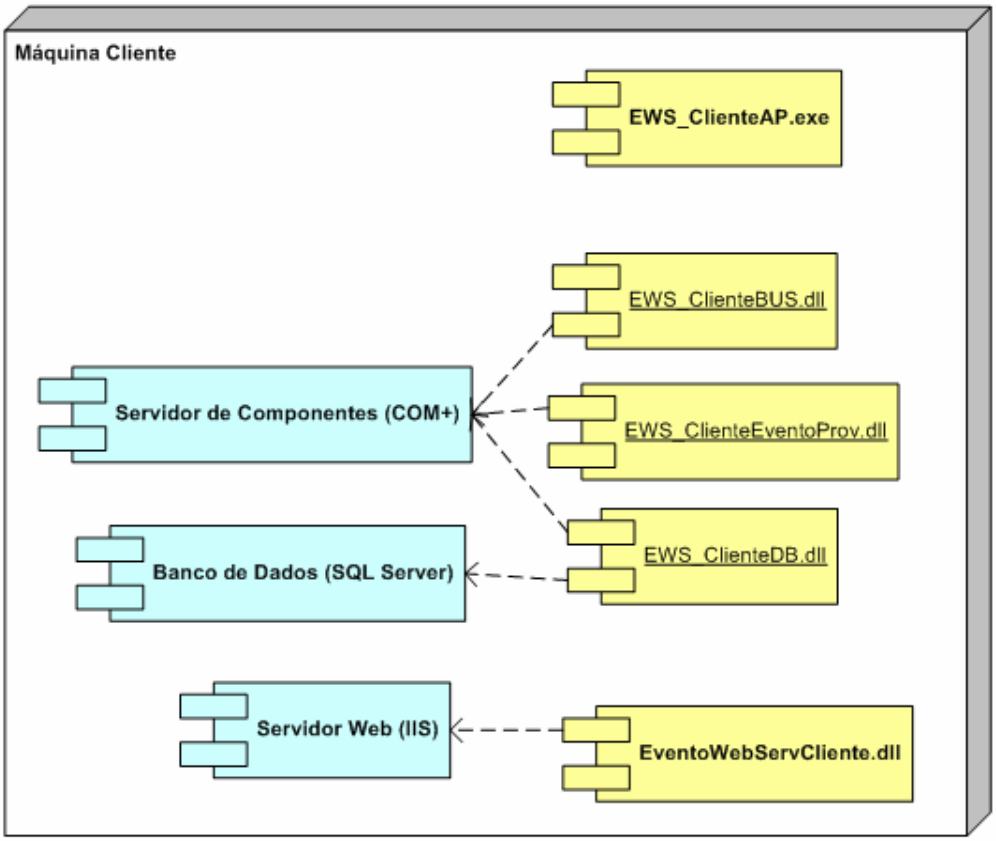

Figura 62 - Diagrama de Implantação do Módulo Cliente

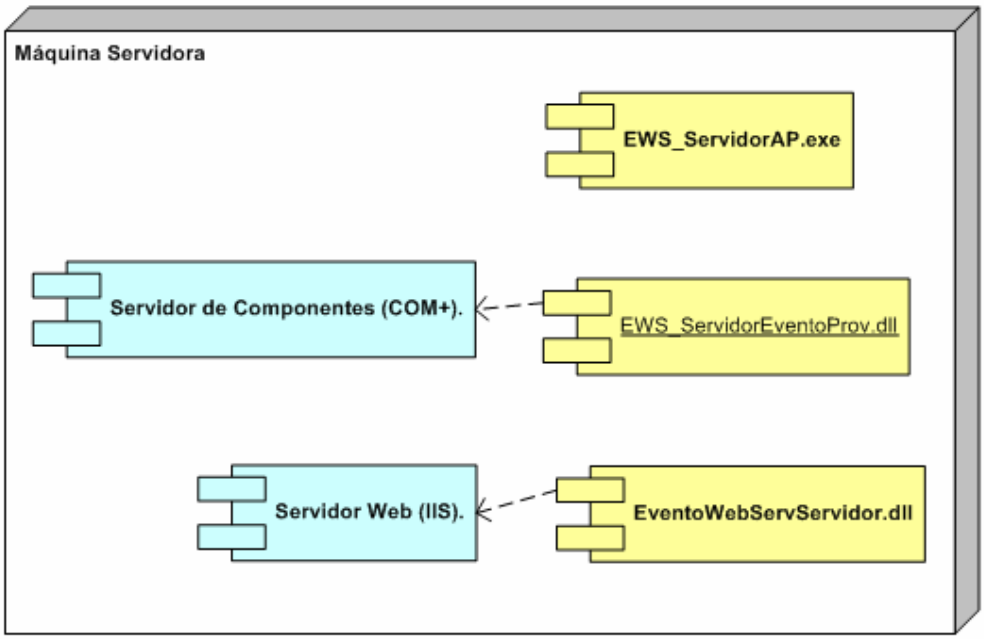

Figura 63 - Diagrama de Implantação do Módulo Servidor

3.9 MODELAGEM DO BANCO DE DADOS

A Figura 64 representa a modelagem do banco de dados "*WebServiceEventos*" utilizado para armazenar as informações geradas pelo sistema. A modelagem foi feita utilizando a ferramenta *Microsoft Visio for Enterprise Architects*.

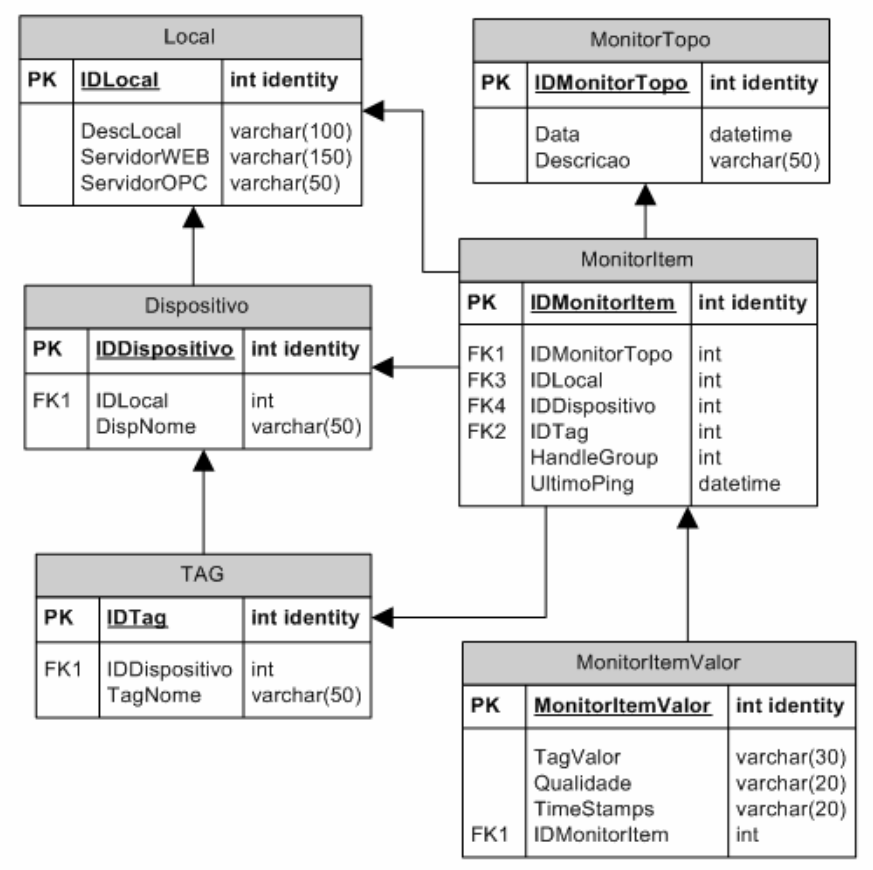

Figura 64 - Modelagem de Banco de Dados "*WebServiceEventos*"

Concluída a modelagem, o *Script* para criação do banco de dados foi gerado e executado pela ferramenta *SQL Query Analyzer*, para que o banco de dados fosse criado fisicamente dentro do SQL Server, juntamente com as tabelas (Figura 65).

```
create database "WebServiceEventos"

go

use "WebServiceEventos"

go

/* Create new table "MonitorItemValor".
create table "MonitorItemValor" (
    "MonitorItemValor" int identity not null,
    "IDMonitorItem" int null,
    "TagValor" varchar(30) null,
    "Qualidade" varchar(20) null,
    "TimeStamps" varchar(20) null)

go

alter table "MonitorItemValor"
    add constraint "MonitorItemValor_PK" primary key ("MonitorItemValor")

go
```

Figura 65 - Comandos para a criação do banco de dados "*WebServiceEventos*"

A Figura 66 demonstra o banco de dados e as tabelas criadas no SQL Server.

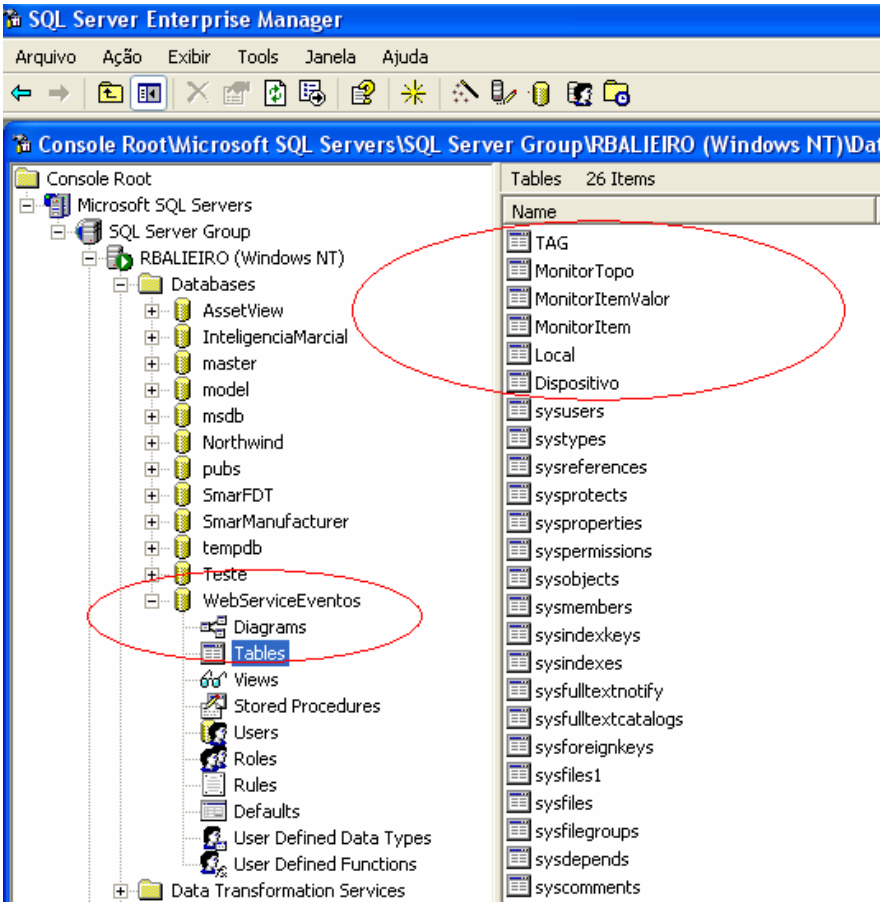

Figura 66 - Banco de dados "*WebServiceEventos*" criado fisicamente no SQL Server

3.10 CODIFICAÇÃO DO PROTÓTIPO

O ambiente utilizado para o desenvolvimento do protótipo foi o *Visual Studio 2005* (Figura 67). O *Visual Studio* faz parte da plataforma .NET da Microsof, um conjunto de tecnologias que permitem o desenvolvimento e execução de aplicações. Os códigos gerados para o .NET podem ser executados em qualquer dispositivo ou plataforma que possua um *.NET Framework*. Para o caso específico deste trabalho, que utiliza o *Visual Studio 2005*, os códigos devem ser executados com o *.NET Framework 2.0*.

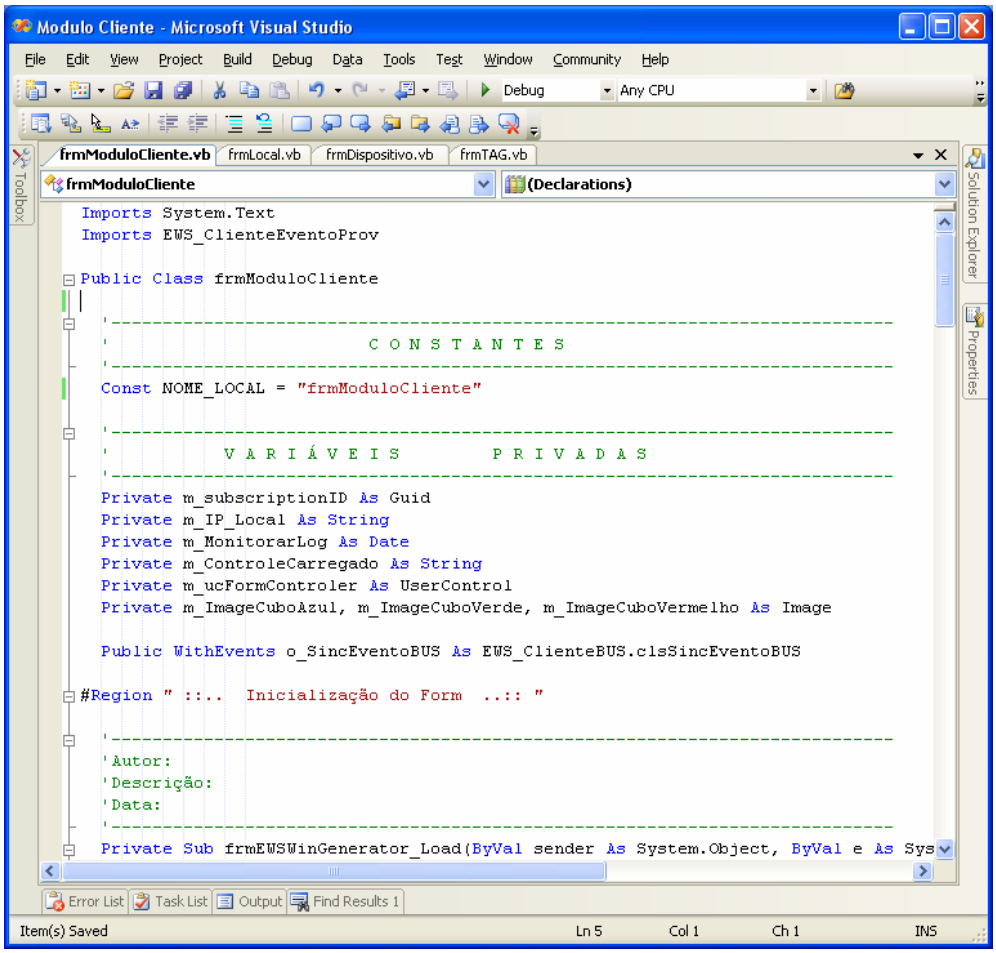

Figura 67 - *Microsoft Visual Studio 2005*

Para o desenvolvimento dos componentes *EWS_ClienteBUS.dll*, *EWSClienteDB.dll* e *EWS_ClienteEventoProv.DLL* das camadas de regra de negócio e de acesso a dados do módulo cliente, foram utilizados recursos do COM+.

O COM+ é uma plataforma da Microsoft para componentes de software, que fornece aos desenvolvedores suporte a transações distribuídas, gerenciamento de instâncias, gerenciamento de concorrência, componentes enfileirados, serviços de evento, entre outros. Para utilizar os serviços de componentes dentro do *Visual Studio 2005*, cria-se uma referência à biblioteca de códigos chamada "*System.EnterpriseServices*" (Figura 68). Feita a referência à "*EnterpriseServices*", pode-se utilizar a biblioteca na construção do código.

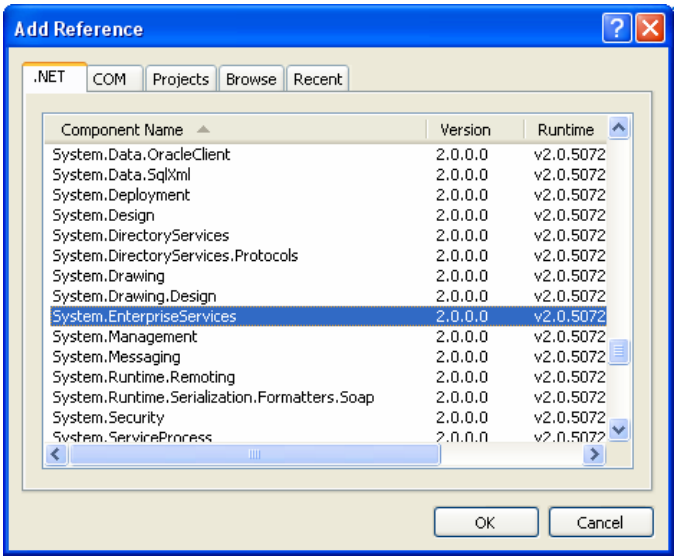

Figura 68 - Referência à biblioteca de código para o COM+

A Figura 69 a seguir ilustra alguns trechos de código utilizando recursos dos serviços de componentes. O trecho de código selecionado no item "1" provê um atalho para a biblioteca do "*EnterpriseServices*". O trecho do item "2" indica que a classe "*clsMonitorBUS*" herda[21] a classe de serviços de componente. O trecho de código do

[21] Herança é o mecanismo pelo qual uma classe pode estender outra classe, aproveitando seus comportamentos (métodos) e estados possíveis (atributos).

item “3” indica quais serviços a classe irá utilizar. Neste caso, serão utilizados os serviços de transação de banco de dados e sincronismo[22] das chamadas ao componente.

A transação de banco de dados permite que se faça, por exemplo, a inserção de informações em duas ou mais tabelas como se fosse uma única operação. Se em algum momento da inserção ocorrer um erro, a transação desfaz qualquer operação que já tenha sido feita anteriormente. Caso nenhum erro ocorra, a transação confirma a operação e os dados ficam armazenados definitivamente no banco de dados.

```
busMonitor.vb   dbMonitorItem.vb
(General)
Imports System
Imports System.Data
Imports System.Diagnostics
Imports System.EnterpriseServices                    1
Imports System.EnterpriseServices.ObjectPool

::.. Interface IMonitor ..::

'-----------------------------------------------------
'                          C L A S S E
'-----------------------------------------------------
<Transaction(TransactionOption.Required), _
Synchronization(SynchronizationOption.Required)> _   3
Public Class clsMonitorBUS
  Inherits ServicedComponent                         2
  Implements IMonitor

  '-----------------------------------------------------
  '                    C O N S T A N T E S
  '-----------------------------------------------------
  Const NOME_LOCAL = "clsMonitorBUS"

  '-----------------------------------------------------
  '          V A R I Á V E I S        P R I V A D A S
  '-----------------------------------------------------
  Private m_tabParam As New DataTable
```

Figura 69 - Código para utilização de serviços de componentes

A Figura 70 ilustra detalhes da classe “*clsSincEventoBUS*”, responsável por receber eventos enviados pelo módulo servidor. Esta classe recebeu o atributo “*SynchronizationOption.Required*”, e isto significa que todas as informações que forem enviadas para esta classe entrarão em uma fila e serão processadas uma após a outra.

[22] O serviço de sincronização providencia um enfileiramento das chamadas ao componente impedindo que mais de uma chamada entre no componente ao mesmo tempo.

```
Imports System.EnterpriseServices

'=================================================
'Autor: Ricardo Luis Balieiro
'Descrição:
'Data:
'=================================================
<Synchronization(SynchronizationOption.Required)> _
Public Class clsSincEventoBUS
  Inherits ServicedComponent
  Implements EWS_ClienteEventoProv.IClienteEventoProv

  '-------------------------------------------------
  '                        C O N S T A N T E S
  '-------------------------------------------------
  Const NOME_LOCAL = "clsSincEventoBUS"
```

Figura 70 - Sincronismo de informações

Esse enfileiramento é necessário porque pode haver mais de um Módulo Servidor enviando dados para o Módulo Cliente (Figura 71). Se este procedimento não tivesse sido implementado, dois eventos vindos de Módulos Servidores diferentes poderiam chegar ao mesmo tempo no Módulo Cliente e seus dados seriam misturados indevidamente.

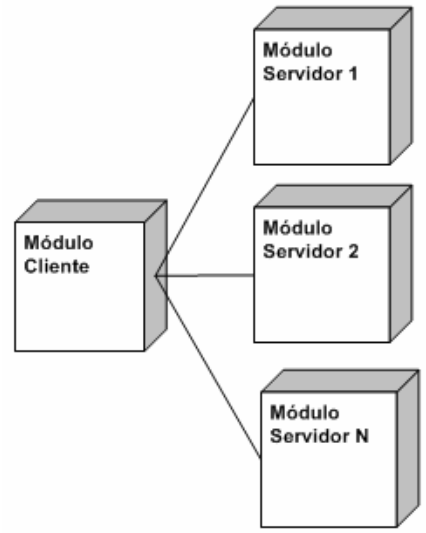

Figura 71 - Módulo cliente recebendo dados de vários módulos servidores

3.11 TELAS DO PROTÓTIPO

As telas do protótipo desenvolvidas neste trabalho são descritas com o auxílio das figuras desta seção, assim como o modo de operação das funções disponíveis em cada tela.

3.11.1 TELA “MÓDULO CLIENTE”: INICIANDO O SISTEMA

A Figura 72 exibe a tela principal do Módulo Cliente. Através desta tela é possível acessar as demais telas do sistema, como será descrito nas subseções a seguir. Ao abrir a tela principal, o sistema efetua uma verificação para confirmar se o *Web Service* e o sistema de eventos estão funcionando corretamente. O estado de funcionamento destes dois componentes é indicado pelos ícones ilustrados na Figura 73. Se o sistema estiver funcionando corretamente, esses ícones são mostrados com a cor verde; caso contrário, os ícones serão mostrados na cor vermelha.

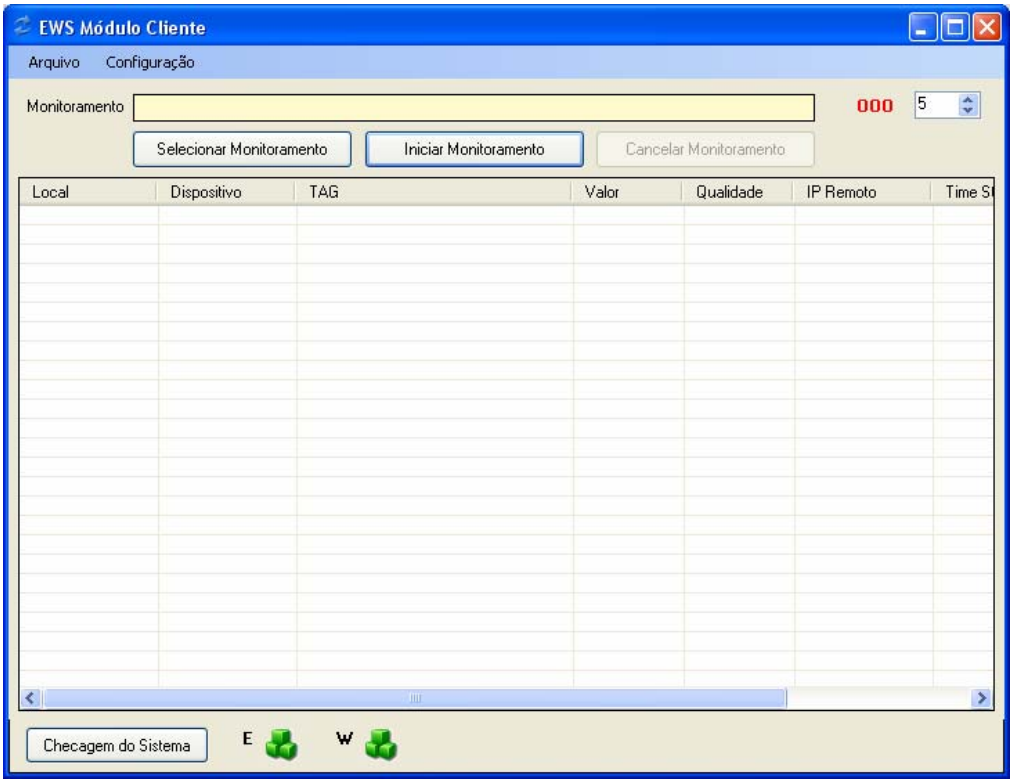

Figura 72 - Tela principal do Módulo Cliente

O botão "*Checagem do Sistema*" permite que a verificação de funcionamento do sistema seja feita a qualquer momento, para detectar possíveis falhas nos componentes.

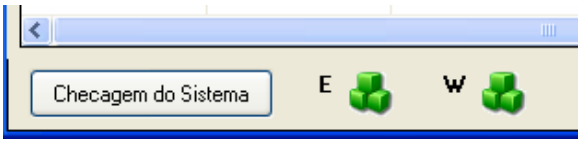

Figura 73 - Controles de checagem de funcionamento do sistema

3.11.2 MENUS

As opções dos menus permitem ao usuário navegar pelas telas de configuração do sistema, conforme ilustra a Figura 74. O menu "*Arquivo*" possui a opção para "*Sair*" do sistema. O menu "*Configuração*" permite acessar as telas para cadastro das informações necessárias para o funcionamento do sistema.

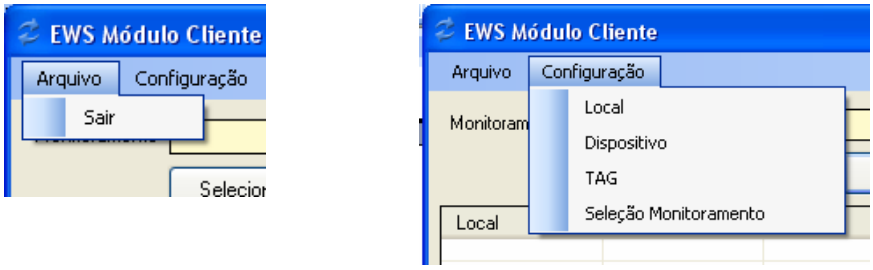

Figura 74 - Menus do sistema do Módulo Cliente

Para efetuar as configurações, é necessário seguir a ordem de chamada das telas conforme exibida nas opções do menu "Configuração". O sistema verifica a execução da seqüência correta de chamadas de telas, e não prossegue com a configuração quando, por exemplo, um usuário tenta cadastrar uma "*TAG*" sem ter previamente cadastrado um "*Dispositivo*".

3.11.3 TELA "LOCAL"

A Figura 75 mostra a tela de manutenção de informações de Locais. Nos campos de edição na parte inferior da tela, o usuário digita as informações e clica no botão "*Salvar*" para confirmar a operação. Então, os dados são salvos no banco de dados e exibidos na tabela de informações já cadastradas. Toda vez que esta tela é aberta, as informações cadastradas para o local são exibidas automaticamente na tabela. Ao clicar e selecionar alguma célula da tabela, as informações são copiadas para os campos de edição para poderem ser alteradas, e os botões "*Atualizar*" e "*Excluir*" são habilitados. O botão "*Fechar*", como o próprio nome diz, fecha o aplicativo de configuração de local.

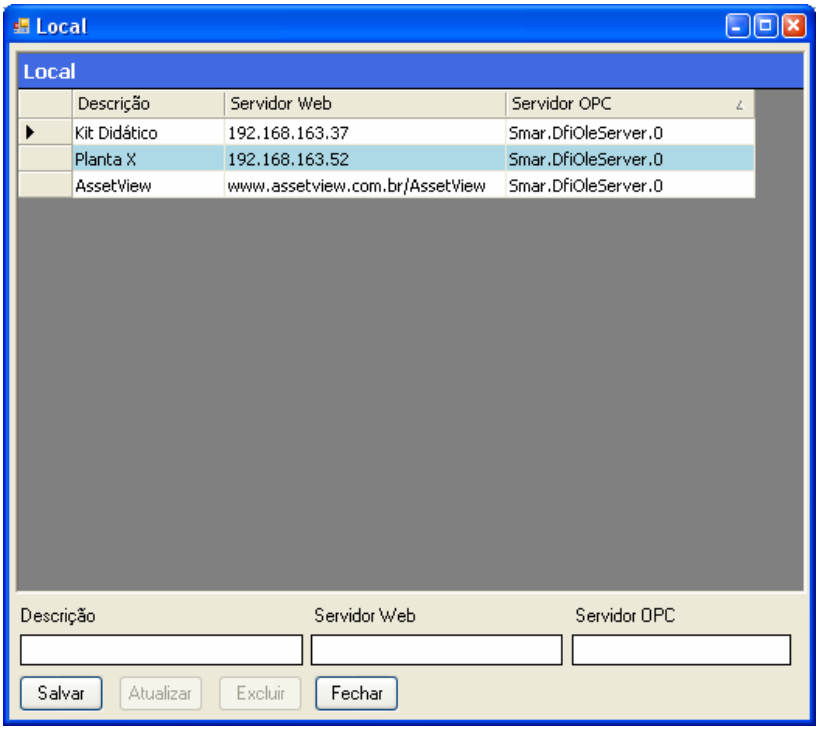

Figura 75 - Tela de manutenção de "Local"

3.11.4 TELA "DISPOSITIVO"

A Figura 76 representa a tela de manutenção de informações de Dispositivos de Campo. O funcionamento desta tela é semelhante ao da tela "Local", porém na parte inferior da tela "Dispositivo" existe a opção "Local" que exibe a lista de locais cadastrados anteriormente, para que o usuário vincule o dispositivo que está sendo cadastrado a um local. Após digitar e salvar as informações, elas são exibidas na tabela de dispositivo.

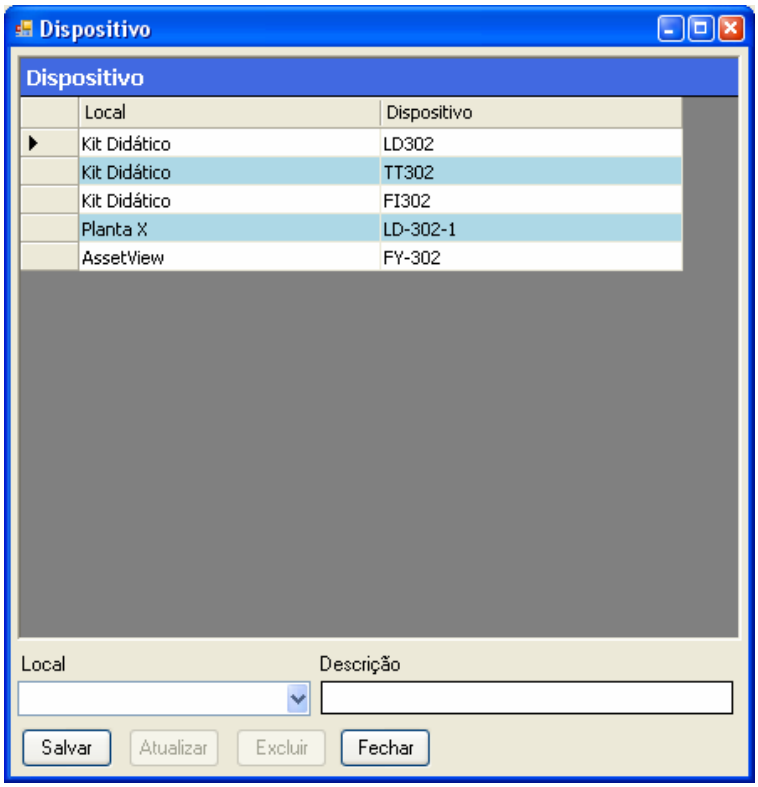

Figura 76 - Tela de manutenção de "Dispositivo de Campo"

Os procedimentos para alterar ou excluir informações de dispositivos de campo são semelhantes aos procedimentos descritos para a tela de "Local".

3.11.5 TELA "TAG"

A Figura 77 exibe a tela de manutenção de informações de tags. O funcionamento desta tela é semelhante às telas anteriores, em particular para os procedimentos de alteração ou exclusão de informações. Na parte inferior da tela "Tag" existe a opção "Dispositivo" que lista os dispositivos cadastrados anteriormente no sistema, para que o usuário vincule a tag que está sendo configurada a um dispositivo. Após digitar e salvar as informações, elas são exibidas na tabela de tags.

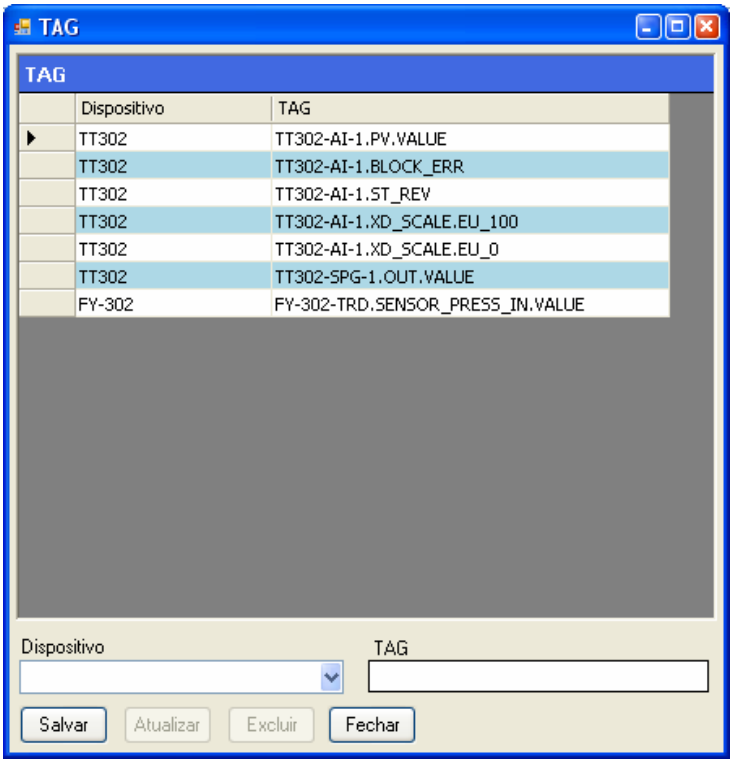

Figura 77 - Tela de manutenção de "Tag"

3.11.6 TELA "SELEÇÃO PARA MONITORAMENTO"

A Figura 78 exibe a tela de "Seleção Para Monitoramento". É necessário que as informações de local, dispositivos e tags já tenham sido cadastradas no sistema para que esta tela funcione corretamente. Para criar uma seleção que será usada no monitoramento, o usuário primeiramente informa o nome da seleção no campo "*Descrição*" e clica no botão "*Salvar*". Esta operação cadastra o nome da seleção no banco de dados e libera as opções para escolha na parte inferior da tela.

Nas opções para escolha, o usuário seleciona as informações para o monitoramento e clica no botão "*Inserir Item*" para confirmar a operação. Este procedimento é executado tantas vezes quanto o usuário achar necessário para completar a lista que será monitorada. O botão "*Excluir Item*" exclui um item da seleção.

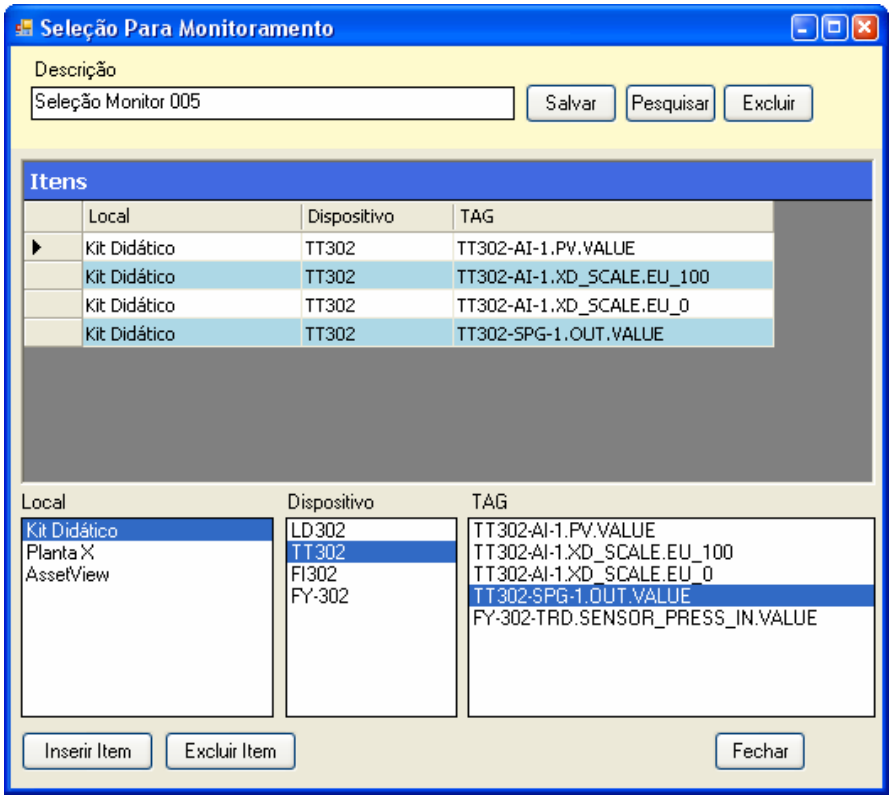

Figura 78 - Tela de manutenção de "Seleção Para Monitoramento"

O botão "*Pesquisar*" na parte superior da tela permite abrir a tela que lista todas as seleções já cadastras no sistema. Esta tela é descrita na subseção 3.11.7, Tela "LISTA MONITORAMENTO". O botão "*Excluir*" exclui uma seleção já cadastrada.

3.11.7 TELA "LISTA MONITORAMENTO"

A Figura 79 ilustra a tela "Lista Monitoramento" que permite ao usuário visualizar os monitoramentos já cadastrados. Esta tela permite também que uma seleção seja escolhida para ser monitorada. Deve-se ressaltar que, para abrir esta tela, é necessário clicar no botão "*Pesquisar*" na parte superior da tela de "Seleção Para Monitoramento".

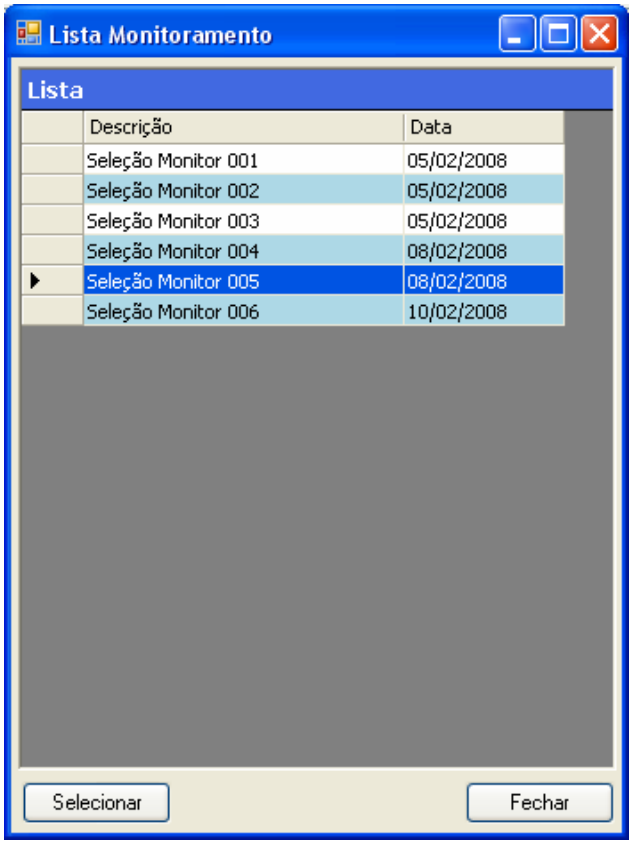

Figura 79 - Tela de seleção de monitoramento

3.11.8 TELA 'MÓDULO CLIENTE': INICIAR MONITORAMENTO

Para iniciar o monitoramento, o usuário deve escolher uma das seleções cadastradas anteriormente, clicando no botão "*Selecionar Monitoramento*" (Figura 80) para abrir a tela "Lista Monitoramento", descrita na subseção anterior. Após escolher uma seleção, o usuário clica no botão "*Iniciar Monitoramento*" para que as tags passem a ser monitoradas. Enquanto este procedimento de monitoramento é executado, o botão "*Cancelar Monitoramento*" fica habilitado.

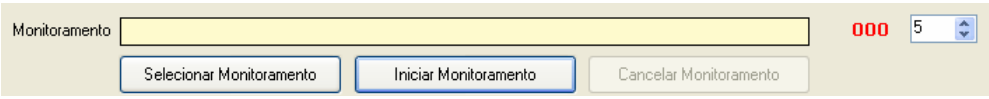

Figura 80 - Controles do sistema de monitoramento

No canto superior direito da tela principal do Módulo Cliente, também ilustrado na Figura 80, é possível alterar o intervalo de tempo em que o sistema atualiza a exibição dos dados na tela. É importante observar que esta opção de tempo controla somente o intervalo de tempo em que o sistema obtém os dados do banco de dados e exibe as informações na tela. Esta opção não tem nenhum controle sobre o intervalo de tempo em que o Módulo Cliente recebe as informações do Módulo Servidor. Como descrito anteriormente nos casos de uso, o Módulo Servidor envia as informações para o Módulo Cliente através de *eventos* e o tempo de envio destes eventos são determinados pelo Servidor OPC.

As informações de retorno enviadas pelo Módulo Servidor e armazenadas no banco de dados são exibidas na tela do formulário. As Figuras 81 e 82 mostram as informações contidas nesta tabela.

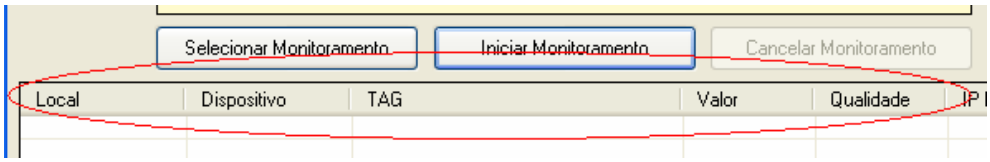

Figura 81 - Tabela de exibição dos dados monitorados - Parte "A"

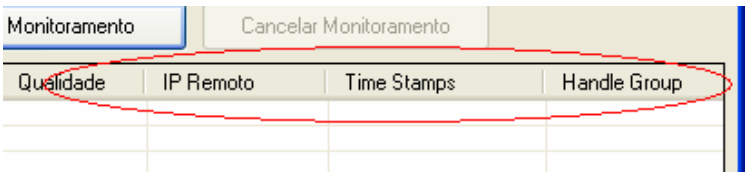

Figura 82 - Tabela de exibição dos dados monitorados - Parte "B"

3.11.9 RECEBENDO INFORMAÇÕES DE MONITORAMENTO

Quando o monitoramento é iniciado, os dados são exibidos na tabela com a cor de fundo vermelha (Figura 83). O fundo vermelho indica que ainda não houve nenhum retorno do Módulo Servidor, nem a verificação de funcionamento da comunicação entre

os módulos. Após receber o sinal indicando que a comunicação está funcionando corretamente, o fundo vermelho é removido. Se em algum momento do monitoramento ocorrer perda da comunicação, a cor de fundo da tabela volta para a cor vermelha.

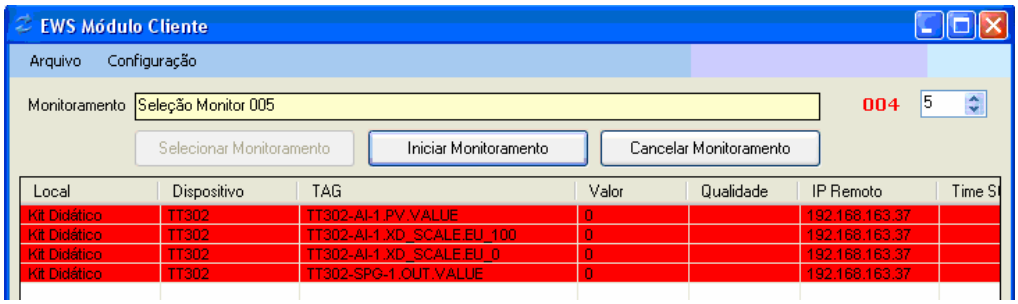

Figura 83 - Aguardando a comunicação entre Módulo Servidor e Módulo Cliente

A Figura 84 exibe os dados monitorados com a cor do texto em vermelho, indicando que ainda não foi retornado nenhum valor pelo Módulo Servidor.

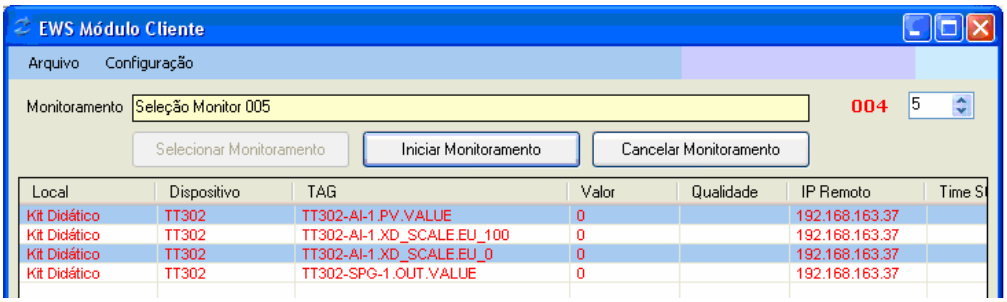

Figura 84 - Confirmação da comunicação e aguardando dados enviados pelo Módulo Servidor

Assim que os primeiros valores são recebidos no Módulo Cliente, o texto na tabela passa a ser mostrado na cor preta (Figura 85).

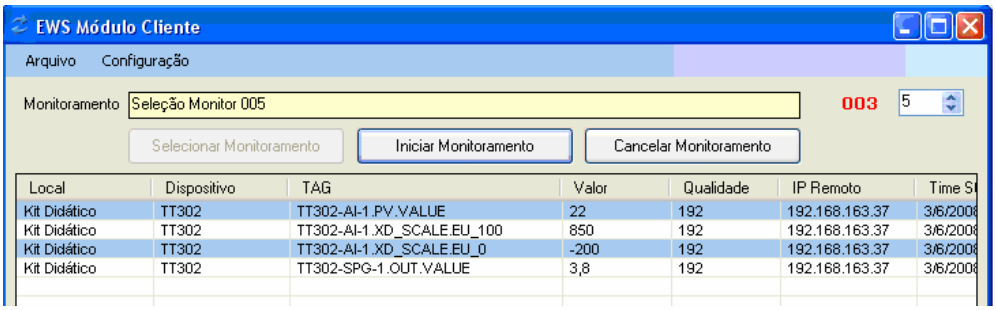

Figura 85 - Confirmação de recebimento de valores enviados pelo Módulo Servidor

3.11.10 TELA "MÓDULO SERVIDOR"

A tela do Módulo Servidor é ilustrada na Figura 86. Quando o Módulo Servidor é iniciado, o sistema checa o sistema de eventos e de *Web Services*. Se os dois sistemas estiverem operando normalmente, os dois ícones na parte inferior da tela ficam na cor verde; caso contrário, os ícones são mostrados na cor vermelha. O botão "*Checagem do Sistema*" permite que a verificação de funcionamento seja feita a qualquer momento, para detectar possíveis falhas nos componentes.

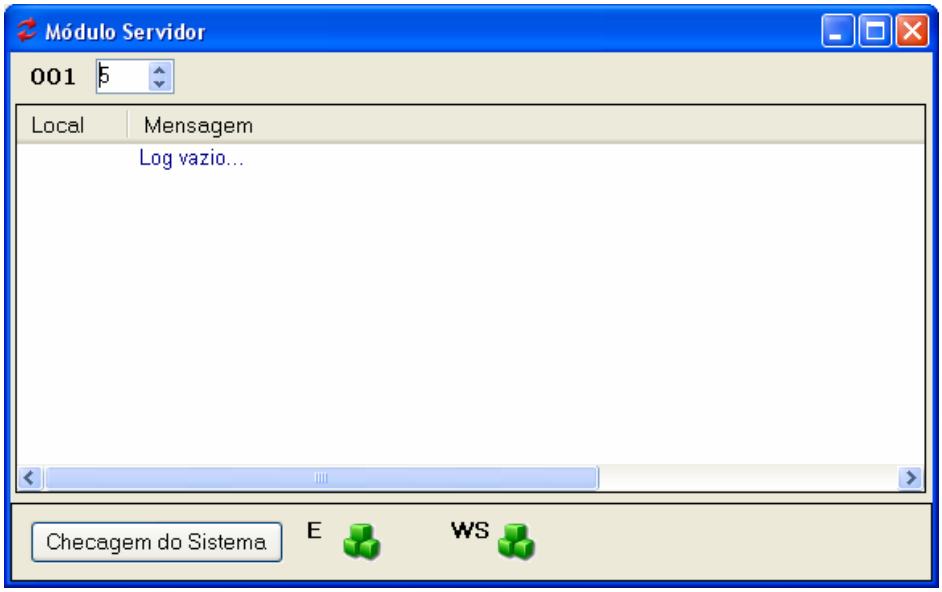

Figura 86 - Tela principal do Módulo Servidor

A tabela no centro da tela exibe mensagens referentes à operação do sistema, como por exemplo, quais tags chegaram para serem monitoradas, quais informações foram enviadas para o Módulo Cliente, falhas ocorridas durante a operação, etc. O intervalo de tempo em que estas mensagens são exibidas na tela é configurado no controle de tempo no canto superior esquerdo da tela.

3.11.11 TELAS WEB SERVICES

Os *Web Services* desenvolvidos no *Visual Studio* para os Módulos Cliente e Servidor têm suas páginas ilustradas nas figuras a seguir. A Figura 87 exibe a página do *Web Service* "*clsEventoWebServCliente*" do Módulo Cliente, com todos os métodos disponíveis para uso.

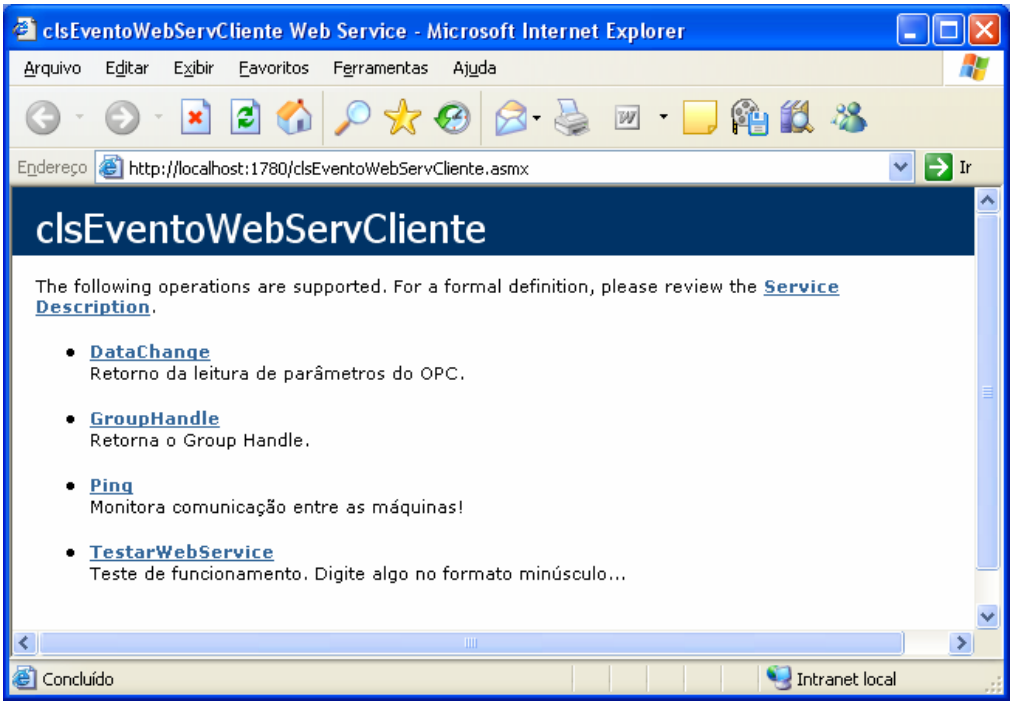

Figura 87 - *Web Service* do Módulo Cliente

A Figura 88 ilustra o *Web Service* "*clsEventoWebServServidor*" do Módulo Servidor.

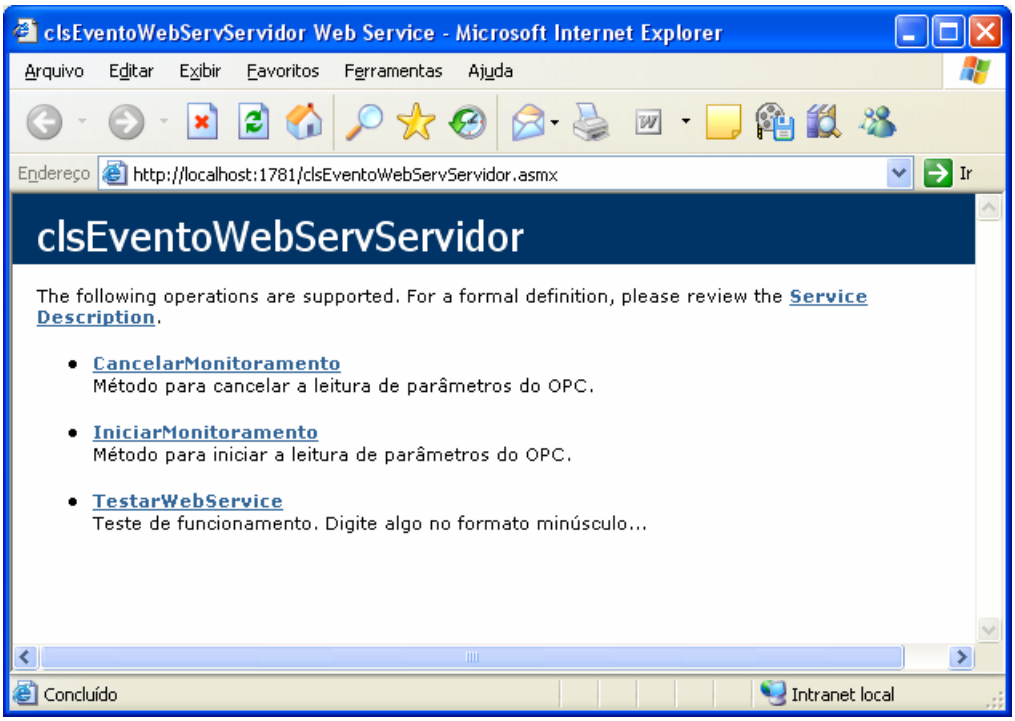

Figura 88 - *Web Service* do Módulo Servidor

3.12 CONFIGURAÇÕES PARA TESTE DO PROTÓTIPO

Uma estrutura de teste foi montada para testar o protótipo. Esta estrutura é composta por diversos módulos: Servidor de Componentes, Servidor Web, Servidor OPC e Kit Didático. As subseções a seguir descrevem as configurações necessárias que foram efetuadas em cada módulo para o correto funcionamento do protótipo.

3.12.1 SERVIDOR DE COMPONENTES

Os componentes desenvolvidos foram instalados nos *Serviços de Componente* do sistema operacional. A Figura 89 mostra os componentes do Módulo Cliente responsáveis pelas regras de negócio (indicados no item 1) e geração de eventos (indicados pelo item 2).

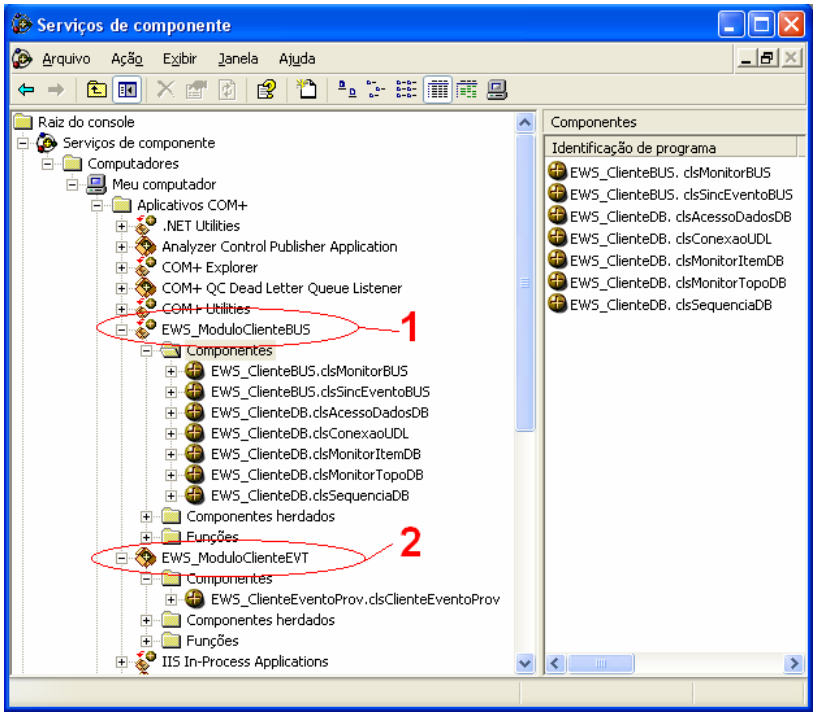

Figura 89 - Componentes do Módulo Cliente instalados nos Serviços de Componente

A Figura 90 mostra o componente de geração de eventos do Módulo Servidor.

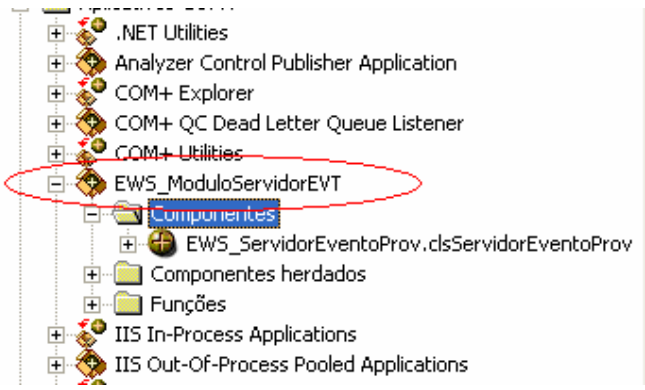

Figura 90 - Componente do Módulo Servidor instalado nos Serviços de Componente

3.12.2 SERVIDOR WEB

Para executar os *Web Services*, foi necessário instalar o Servidor Web nas máquinas cliente e servidor, porque o Servidor Web não é instalado automaticamente quando o sistema operacional Windows é instalado. Apesar de existirem vários Servidores Web disponíveis no mercado, tais como Apache, IIS e Enterprise Server da

Iplanet, foi escolhido o IIS desenvolvido pela Microsoft pelo fato do IIS ser o mais compatível com a tecnologia ASP.NET, que foi a tecnologia utilizada no desenvolvimento dos *Web Services*. Tanto na máquina cliente como na máquina servidora, ambas operando com Windows XP Professional e Service Pack 2, foi instalado o IIS 5.1.

Outro ponto importante para o funcionamento dos *Web Services* foi a instalação da versão Framework 2.0 nas máquinas, utilizada pelo ASP.NET 2005, onde foram desenvolvidos os *Web Services*. Na máquina cliente e na máquina servidora, os arquivos dos *Web Services* foram colocados em subdiretórios do diretório padrão do IIS, no caminho "C:\Inetpub\wwwroot". O nome da aplicação define o nome do subdiretório: na máquina cliente os arquivos foram colocados no diretório "C:\Inetpub\wwwroot\EventoWebServCliente", enquanto que na máquina servidora os arquivos ficaram em "C:\Inetpub\wwwroot\EventoWebServServidor".

Feito isso, foram criados os diretórios virtuais[23] da aplicação (também chamados de Web Sites) dentro do IIS. Quando um diretório virtual é criado no IIS, ele fica marcado como um aplicativo Web. A Figura 91 mostra o diretório virtual *EventoWebServCliente* criado na máquina cliente. Todos os procedimentos de configuração descritos a seguir foram aplicados da mesma forma para o diretório virtual *EventoWebServServidor* na máquina servidora.

[23] Um diretório virtual representa um diretório físico na Web, mas não precisa necessariamente ter o mesmo nome que o diretório físico.

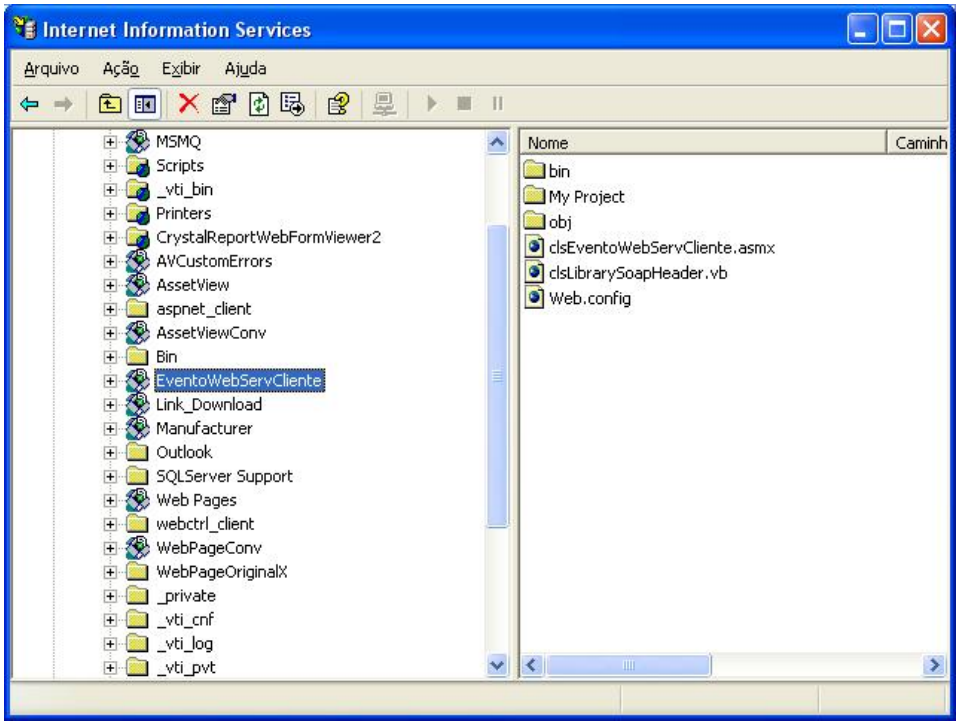

Figura 91 - Criação do diretório virtual (Web Site) do *Web Service* cliente no IIS

As propriedades do diretório virtual criado foram configuradas conforme a Figura 92. Apenas a permissão de leitura ("*Ler*") foi habilitada para o Web Site, desabilitando as permissões de "*Gravar*" e "*Pesquisa no diretório*". A opção "*Pesquisa no diretório*" permite a exibição de uma lista com todos os nomes de documentos contidos no diretório quando um usuário acessa o Web Site. As permissões "*Ler*" e "*Somente Script*" são suficientes para que a aplicação esteja totalmente funcional via Internet.

Nas propriedades de "*Proteção do aplicativo*", a opção "Média (em Pool)" foi selecionada, para permitir que o processo seja executado em um ambiente isolado.

Figura 92 - Configurações do diretório virtual do *Web Service* cliente

Outra configuração necessária para o funcionamento do Web Site é a configuração da versão do ASP.NET utilizado no desenvolvimento do *Web Service*. Esta configuração é feita na pasta ASP.NET, selecionando a versão ASP.NET 2.0 conforme ilustrado na Figura 93. Caso este passo não seja executado, o IIS usará a configuração padrão feita automaticamente durante a criação do Web Site, ou seja, o ASP.NET 1.1. Então, se o Web Site possui arquivos da versão ASP.NET 2.0 mas está configurado para o ASP.NET 1.1, o IIS retornará o erro 404 quando o *Web Service* for requisitado.

Figura 93 - Configurações do ASP.NET 2.0 para o *Web Service* cliente

3.12.3 SERVIDOR OPC

A conexão entre o DFI302 e o protótipo é feita utilizando o Servidor OPC da Smar chamado *OPC DFI OLE Server*. Este Servidor OPC atende à especificação OPC Data Access 2.05 e é certificado pela OPC Foundation. A instalação do *OPC DFI OLE Server* foi feita pelo pacote de software *System302* da Smar em uma estação com sistema operacional Windows Server 2003. O *System302* contém um aplicativo chamado *Server Manager* que permite o gerenciamento de vários Servidores OPC.

Para que o Servidor OPC funcione corretamente, é necessário configurar o cartão de interface de rede, ou NIC, informando o endereço de IP da placa de rede instalada no servidor (Figura 94).

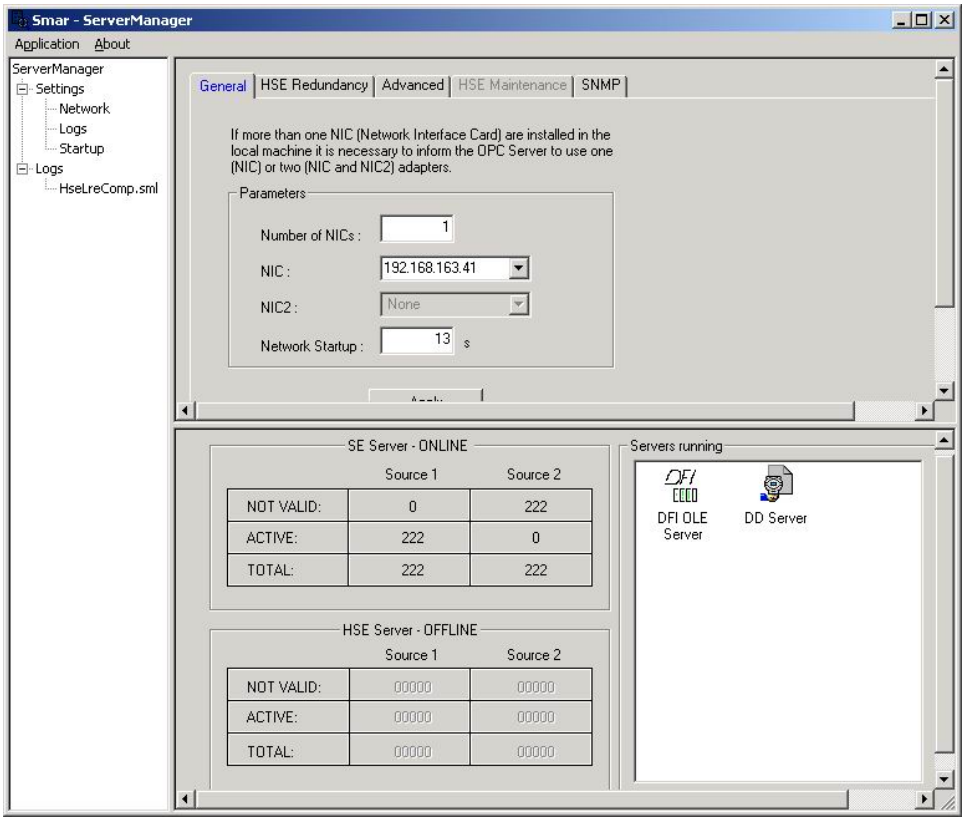

Figura 94 - Tela do *Server Manager*

3.12.4 KIT DIDÁTICO SMAR

A rede de comunicação foi estruturada com o kit didático Smar (Figura 95), utilizando protocolo Foundation™ Fieldbus. O kit didático Smar é composto por um sistema modular DFI302 (*Fieldbus Universal Bridge*) que possui interface de comunicação para quatro canais H1 Foundation™ Fieldbus e a rede *High Speed Ethernet* (HSE - 100 Mbps). O *Syscon* permite que toda a configuração do sistema seja descarregada nos equipamentos de campo e no próprio DFI302.

Foram utilizados os seguintes módulos na configuração do kit didático:

- **DF50**: fonte de alimentação;
- **DF60**: fonte de alimentação DC para Foundation™ Fieldbus;
- **DF52**: fonte de alimentação AC para Foundation™ Fieldbus;

- **DF49:** impedância para fonte de alimentação Foundation™ Fieldbus - duas portas;
- **DF51**: controlador com uma porta 10 Mbps e quatro canais H1 Foundation™ Fieldbus;
- **HI302:** interface Hart/Foundation™ Fieldbus.

Os equipamentos de campo utilizados no kit didático fazem parte da linha 302 da Smar. São eles:

- **LD302 - Transmissor de Pressão Fieldbus Foundation:** transmissor para medida de pressão diferencial, absoluta e manométrica, nível e vazão;
- **TT302 - Transmissor de Temperatura Fieldbus Foundation:** transmissor apropriado para medições de temperatura usando termoresistências ou termopares, mas pode também aceitar outros sensores que gerem resistência ou milivoltagem, tais como pirômetros, células de carga, indicadores de posição, etc;
- **FI302 - Conversor de Fieldbus para corrente:** conversor que faz a interface de uma rede *Fieldbus* com uma válvula de controle e outros atuadores. Fornece uma saída de 4-20 mA proporcional a entrada recebida de um sistema *Fieldbus*.

Para simular um sensor de temperatura tipo PT100, um potenciômetro foi acoplado ao TT302. Variando o potenciômetro tem-se a simulação de temperatura entre 100ºC a 500ºC. Uma bomba de pressão manual conectada ao LD302 simula o controle

de nível e vazão. As saídas do TT302 e do LD302 foram ligadas ao conversor FI302 para simular a saída para um atuador na planta.

Figura 95 - Kit didático Smar

3.12.5 CONFIGURAÇÃO DA REDE INDUSTRIAL

As configurações básicas dos equipamentos do kit didático são feitas através dos blocos funcionais e pré-definidas pelo fabricante Smar. Os blocos descrevem as funções disponíveis em um dado equipamento e definem como estas funções podem ser acessadas. Todo bloco funcional tem um nome (tag) definido pelo usuário que deve ser único na rede Foundation™ Fieldbus. A configuração da estratégia de controle da rede Foundation™ Fieldbus foi feita com o software de configuração da Smar, chamado *Syscon*. O *Syscon* permite configurar os equipamentos, definir suas finalidades e grandezas, e criar a estratégia de controle.

Um novo arquivo de projeto foi criado no *Syscon*. Para agilizar a configuração do projeto, foi utilizado o procedimento do *Syscon* que executa um serviço de busca na

rede Foundation™ Fieldbus e identifica os controladores e equipamentos que estão conectados a rede, criando suas representações no projeto. Para executar este procedimento, a comunicação foi iniciada, o *Syscon* passou a ser executado no modo *Online* e a opção "*Upload*" foi selecionada para a rede fieldbus. O controlador da rede e todos os equipamentos conectados aos canais do controlador, assim como seus blocos e parâmetros pré-definidos, foram automaticamente instanciados no projeto.

O módulo DF51 possui quatro canais de comunicação que podem ser configurados independentemente, de acordo com os instrumentos que estão sendo utilizados em cada canal. Para a configuração do kit didático, apenas um canal do módulo DF51 foi utilizado, o canal denominado "Canal Fieldbus 1" (Figura 96).

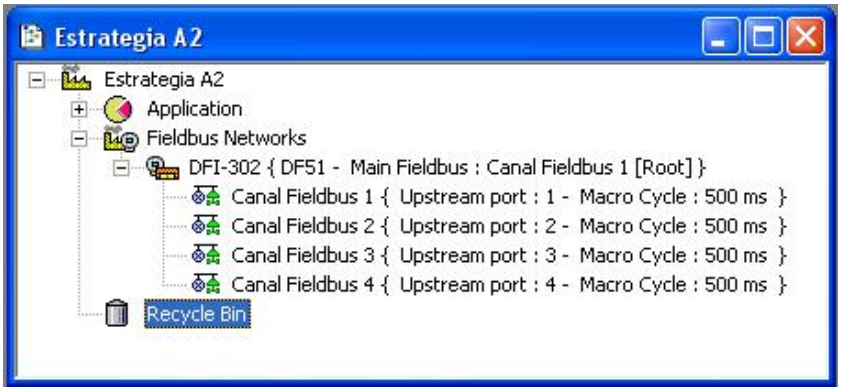

Figura 96 - Configuração do canal Fieldbus

Os equipamentos do kit didático são configurados com um nome padrão definido pelo fabricante, mas estes nomes foram alterados para facilitar a identificação de tags no projeto. Por exemplo, o módulo DF51 foi renomeado com a tag "DFI-302". No canal "Canal Fieldbus 1", foram identificados os três instrumentos conectados a rede, que também foram renomeados como: TT302, FI302 e LD302 (Figura 97).

Considerando-se as características do kit didático, foram criados blocos funcionais que permitiriam variar de alguma forma seus parâmetros e, assim, prover informações úteis para serem coletadas pelo protótipo. É importante observar que todos

os equipamentos possuem blocos fundamentais que devem ser sempre instanciados. São eles: blocos *Resource*, que descrevem as características físicas dos equipamentos; blocos *Transducer*, que contêm parâmetros de calibração e diagnóstico; e blocos *Display*, responsáveis pela formatação da informação mostrada no visor do equipamento.

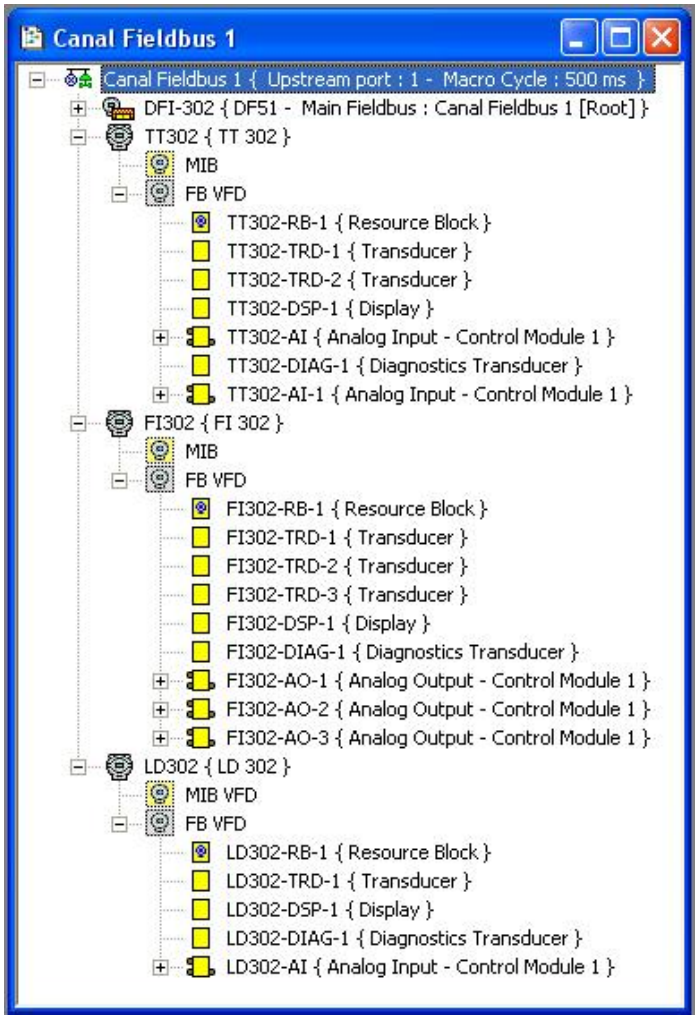

Figura 97 - Configurações dos blocos dos equipamentos

Depois de adicionar todos os blocos necessários, seus parâmetros foram configurados para executar as funções desejadas corretamente. Então, os blocos foram interligados na janela de estratégia, conforme a Figura 98. Como há uma limitação de equipamentos, apenas as saídas dos blocos *Entrada Analógica* (AI) dos equipamentos LD302 e TT302 foram ligadas às entradas dos blocos *Saída Analógica* (AO) do FI302.

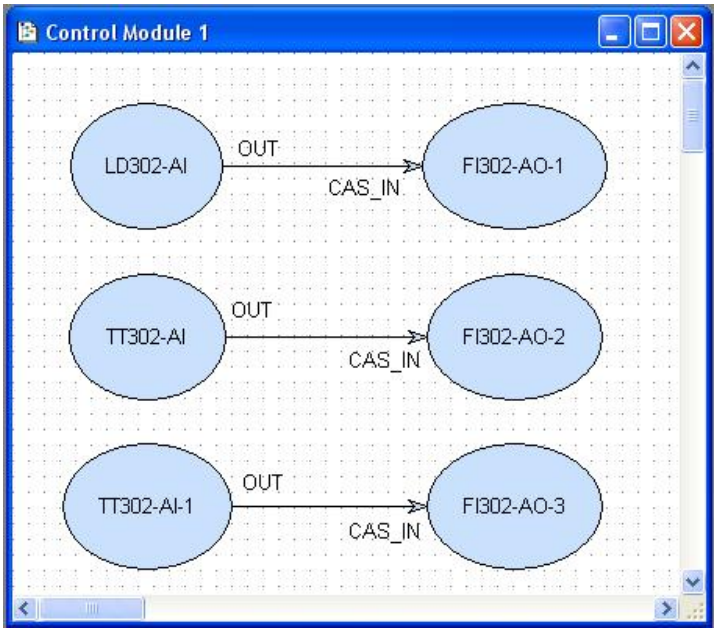

Figura 98 - Interligação dos blocos dos equipamentos

Finalizada a configuração da estratégia de controle, toda a informação do projeto foi descarregada nos equipamentos, executando-se o procedimento "*Download*" do canal de comunicação do DF51. Este procedimento é executando quando o *Syscon* está operando no modo *Online*, ou seja, o software está comunicando com a rede fieldbus. Durante o procedimento, os blocos e parâmetros criados "virtualmente" no arquivo de projeto do *Syscon* são instanciados fisicamente nos equipamentos do kit didático.

Depois que toda a configuração foi descarregada, é possível para alterar variáveis do processo de controle, como temperatura ou pressão, e obter os dados necessários para validar o protótipo.

4 RESULTADOS E DISCUSSÃO DOS TESTES

O objetivo principal dos testes descritos abaixo é validar o funcionamento do protótipo e analisar seu desempenho, contrapondo os resultados obtidos com os requisitos propostos para o protótipo. O primeiro conjunto de testes foi realizado para validar o funcionamento do protótipo. Os resultados do segundo conjunto de testes foram analisados para avaliar o desempenho do protótipo, considerando duas situações: o “caso crítico” de desempenho e o “melhor caso”.

Um caso de desempenho crítico para transferência de dados durante o monitoramento de tags utilizaria uma conexão discada para acessar a Internet; em contrapartida, considera-se que o melhor desempenho do protótipo ocorre quando uma rede local Ethernet é utilizada para monitorar tags.

Antes de executar todos os testes, o relógio do sistema operacional da máquina cliente e da máquina servidora foram sincronizados através do servidor local de rede. Este sincronismo foi definido para obter os tempos de comunicação entre os módulos.

4.1 VALIDAÇÃO DE FUNCIONAMENTO

4.1.1 ESTÁGIO 1: INICIAR O MONITORAMENTO DE TAGS

Esta seção descreve os testes realizados para validar o funcionamento do primeiro estágio do monitoramento de tags. Estes testes consistem em enviar um conjunto de tags do Módulo Cliente para o Módulo Servidor. O Módulo Servidor configura o Servidor OPC para monitorar as tags recebidas, e o Servidor OPC retorna o *Group Handle* relativo às tags que serão monitoradas para o Módulo Servidor. Então, o Módulo Servidor envia o *Group Handle* para o Módulo Cliente e, por fim, o Módulo Cliente armazena o *Group Handle* no banco de dados e atualiza a tela do sistema, indicando se a comunicação e o monitoramento estão funcionando corretamente.

Aplicando os procedimentos acima descritos, o protótipo respondeu dentro do esperado, ou seja, o Módulo Cliente enviou o grupo de tags para serem monitoradas e recebeu o *Group Handle* do Módulo Servidor. Este resultado valida e conclui o primeiro estágio de funcionamento do protótipo.

4.1.2 ESTÁGIO 2: MONITORAMENTO CONTÍNUO

Esta seção descreve os testes realizados para validar o funcionamento do segundo estágio do monitoramento de tags. Para a realização deste estágio, o monitoramento de tags descrito no Estágio 1 deve estar funcionando corretamente. O segundo estágio consiste em monitorar a transmissão dos valores de tags entre o Módulo Servidor e o Módulo Cliente, durante um período de tempo. Este monitoramento inicia-se com o Servidor OPC enviando eventos *Data Change* para o

Módulo Servidor. O Módulo Servidor recebe os valores das tags no evento *Data Change* e os envia para o Módulo Cliente. O Módulo Cliente recebe os valores das tags e os armazena no banco de dados. Um processo secundário no Módulo Cliente se encarrega de exibir os valores atualizados na tela.

Este procedimento foi aplicado ao protótipo e monitorado durante um período de tempo de 30 minutos. Durante este tempo, o protótipo respondeu dentro do esperado, ou seja, as tags enviadas pelo Módulo Servidor foram recebidas e armazenadas no banco de dados no Módulo Cliente. Este resultado valida e conclui o segundo estágio de funcionamento do protótipo.

4.1.3 ESTÁGIO 3: CANCELAR O MONITORAMENTO DE TAGS

Esta seção descreve os testes realizados para validar o funcionamento do terceiro estágio do monitoramento de tags. Para a realização deste estágio, o monitoramento contínuo descrito no Estágio 2 deve estar funcionando corretamente. O terceiro estágio consiste na operação do Módulo Cliente enviando uma solicitação para que o Módulo Servidor pare de monitorar as tags. Ao receber a solicitação, o Módulo Servidor remove as tags do Servidor OPC, para que este servidor pare a monitoração. Ao mesmo tempo, o Módulo Servidor pára de transmitir qualquer informação para o Módulo Cliente.

Aplicando os procedimentos acima descritos, o protótipo também respondeu dentro do esperado, ou seja, o Servidor OPC parou de monitorar as tags e o Módulo Servidor deixou de enviar qualquer informação para o Módulo Cliente. Este resultado valida e conclui o terceiro e último estágio de funcionamento do protótipo.

4.2 ANÁLISE DE DESEMPENHO

Para os testes de desempenho, foi criada uma nova configuração utilizando o bloco funcional "*Setpoint Ramp Generator*" (SPG), ilustrado na Figura 99, foi adicionado ao transmissor de temperatura do kit didático. O bloco SPG gera valores dentro de um intervalo de tempo.

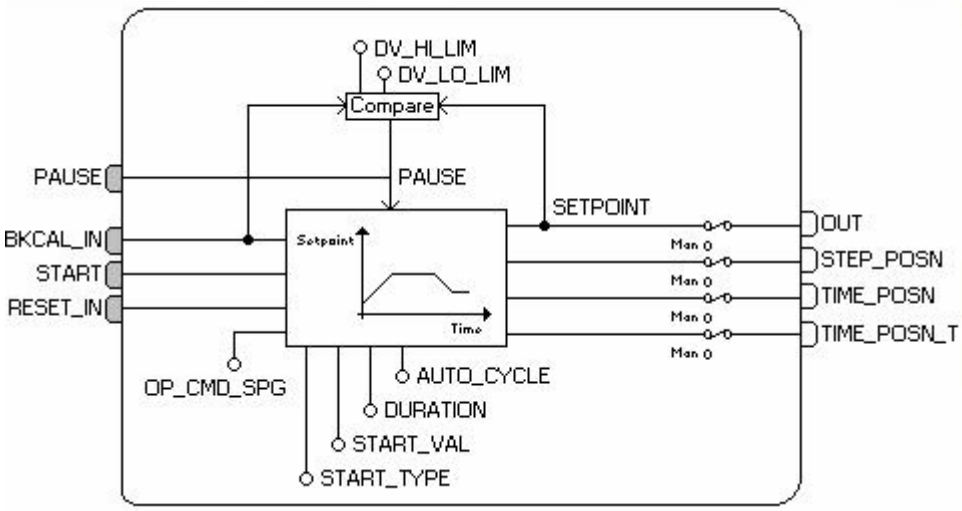

Figura 99 - Bloco "Setpoint Ramp Generator" (SPG)

A principal configuração deste bloco foi feita nos parâmetros "START_VAL" e "DURATION" (Figura 100). Os valores de saída do parâmetro "START_VAL", que são monitorados no parâmetro "OUT.VALUE", foram configurados para variar entre 10 e 110, e em seguida retornar a 10.

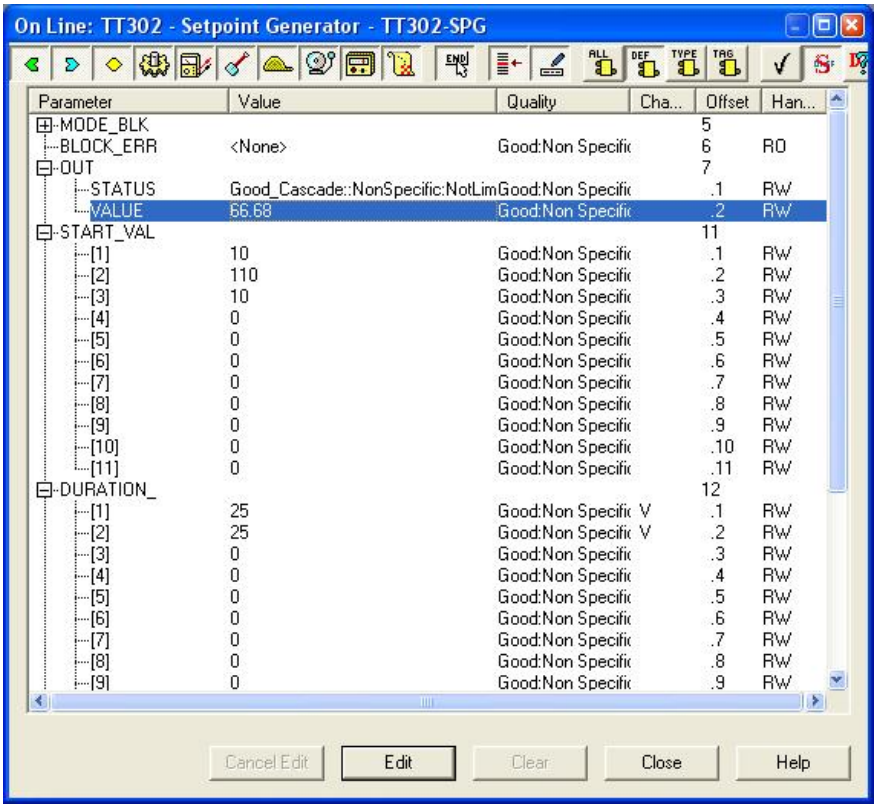

Figura 100 - Configuração do bloco "*Setpoint Ramp Generator*" (SPG)

A duração da variação entre os valores 10-110-10 foi configurada para 25 segundos no parâmetro "DURATION". Esta configuração faz com que sejam gerados, no parâmetro "OUT.VALUE", valores em rampa que vão de 10 a 110 em 25 segundos (Figura 101).

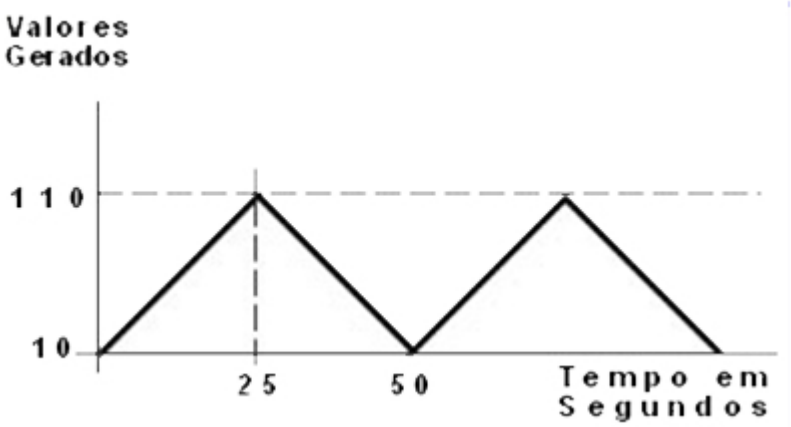

Figura 101 - Valores gerados em "OUT.VALUE"

O *macro-ciclo* da rede Foudation Fieldbus do kit didático também foi configurado. O macro-ciclo é o intervalo de tempo em que toda a parte cíclica se repete na rede. O menor valor calculado para o macro-ciclo foi de 570 milisegundos (Figura 102). Tanto o macro-ciclo quanto o valor configurado para o parâmetro "DURATION" foram definidos com o objetivo de atingir a maior velocidade de geração de valores para o kit didático. Apesar de configurar os valores com o mínimo permitido, a taxa conseguida foi em torno de uma a duas tags por segundo.

Figura 102 - Configuração do macro-ciclo

4.2.1 DESEMPENHO 1: CASO CRÍTICO

Para o teste de caso crítico, foi utilizada uma comunicação através de linha discada entre o Módulo Cliente o Módulo Servidor. O Módulo Cliente utilizou um modem com uma taxa de comunicação de 36 kbps, e a comunicação do modem do Módulo Servidor atingiu a taxa de 40 kbps. A quantidade de eventos *Data Change* gerados pelo Servidor OPC ficou entre um e dois por segundo devido à configuração feita no kit didático, descrita anteriormente.

Quatro tomadas de monitoramento com tempo de 15 minutos foram efetuadas, e nas quatro amostragens os valores não se alteraram. A Tabela 06 demonstra a média dos valores obtidos. Devido às baixas taxas de transmissão, não foi possível transmitir todos os *Data Change* gerados pelo Módulo Servidor.

Tabela 06 - Tempo Médio de Transmissão no Caso Crítico

Período de Amostragem, em segundos	**Quantidade de *Data Change* Gerados**	**Quantidade de *Data Change* Enviados**	**Tempo médio de envio de um *Data Change*, em segundos**
901	1578	358	2,51

A Tabela 07 exibe uma amostragem dos valores de tempo obtidos na transmissão entre o Módulo Servidor e o Módulo Cliente.

Tabela 07 - Tempo de Transmissão no Caso Crítico

***Data Change* Transmitido**	**Servidor**	**Cliente**
1	12:55:46	12:55:51
2	12:55:52	12:55:54
3	12:55:54	12:55:59
4	12:55:59	12:56:02
5	12:56:02	12:56:06
6	12:56:06	12:56:08
7	12:56:08	12:56:10
8	12:56:10	12:56:12
9	12:56:13	12:56:14
10	12:56:14	12:56:17
11	12:56:17	12:56:20
12	12:56:20	12:56:22
13	12:56:22	12:56:25
14	12:56:25	12:56:29
15	12:56:29	12:56:32
16	12:56:33	12:56:36
17	12:56:36	12:56:39
18	12:56:39	12:56:41
19	12:56:42	12:56:44
20	12:56:44	12:56:46
21	12:56:46	12:56:48
22	12:56:48	12:56:50
23	12:56:50	12:56:53
24	12:56:53	12:56:55
25	12:56:55	12:56:57
26	12:56:57	12:56:59

27	12:56:59	12:57:01
28	12:57:01	12:57:03
29	12:57:03	12:57:05
30	12:57:05	12:57:08
31	12:57:08	12:57:10
32	12:57:10	12:57:13
33	12:57:13	12:57:15
34	12:57:15	12:57:18

Com base nos dados obtidos foi gerado o gráfico da Figura 103. Este gráfico ilustra o atraso de tempo utilizado para a transmissão dos *Data Change* entre o Módulo Servidor e o Módulo Cliente.

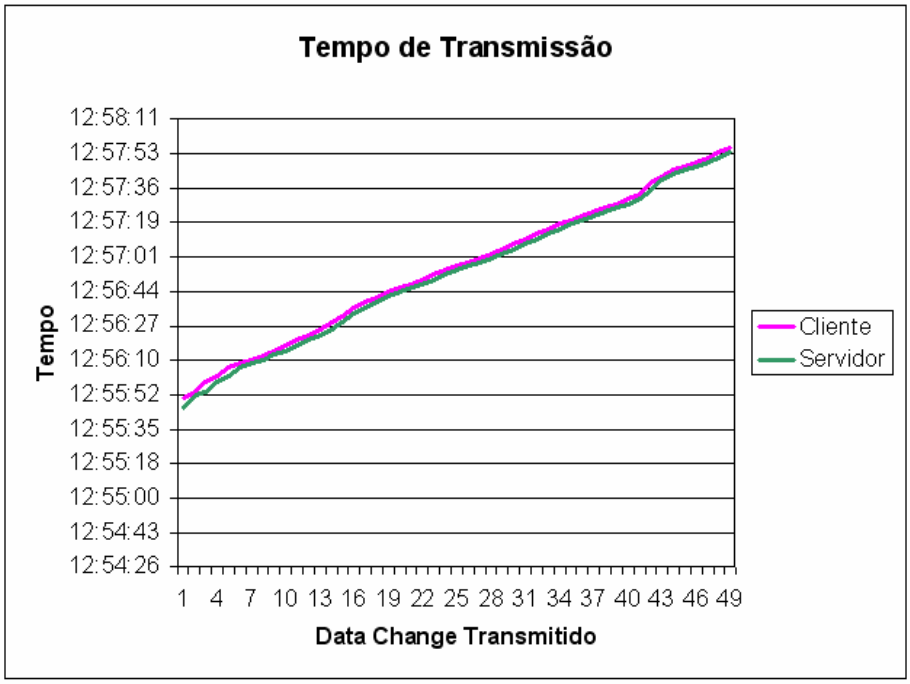

Figura 103 - Gráfico de tempo de transmissão do caso crítico

Novos testes foram executados para o caso crítico utilizando o emulador do Servidor OPC, cujo funcionamento está descrito em detalhes na subseção *4.2.2 Desempenho 2: Melhor Caso* a seguir, sobre os testes de desempenho para o "melhor caso". Primeiramente os testes foram executados com o emulador gerando quatro eventos *Data Change* por segundo. Em seguida, os testes foram feitos com o emulador gerando oito eventos *Data Change* por segundo. Observou-se que a média de

transmissão manteve-se constante em relação aos testes anteriores, ou seja, independente da quantidade de eventos *Data Change* gerados, foi possível transmitir um *Data Change* a cada 2,51 segundos utilizando a linha discada.

A Figura 104 mostra o gráfico referente ao comportamento observado.

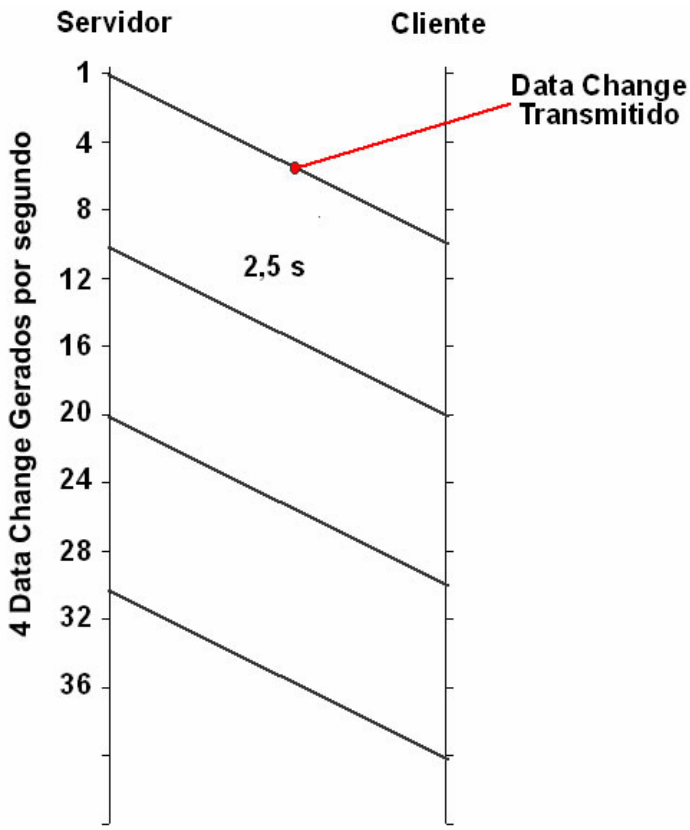

Figura 104 - Analise de desempenho do caso crítico

4.2.2 DESEMPENHO 2: MELHOR CASO

Para o teste de melhor caso, utilizou-se uma rede Ethernet como meio de transmissão com uma taxa de transmissão de 100 Mbps. Neste caso, notou-se que não houve perda de tempo considerável na transmissão entre o Módulo Servidor e o Módulo Cliente. Diante deste resultado, a configuração de teste foi alterada para obter taxas mais rápida na geração de valores, utilizando a característica do Servidor OPC da Smar que permite emular valores para uma tag monitorada. Para isto, basta alterar o arquivo

de configuração do Servidor OPC, chamado “SmarOleServer.ini”. A Figura 105 mostra o arquivo sendo editado e como os parâmetros devem ser configurados.

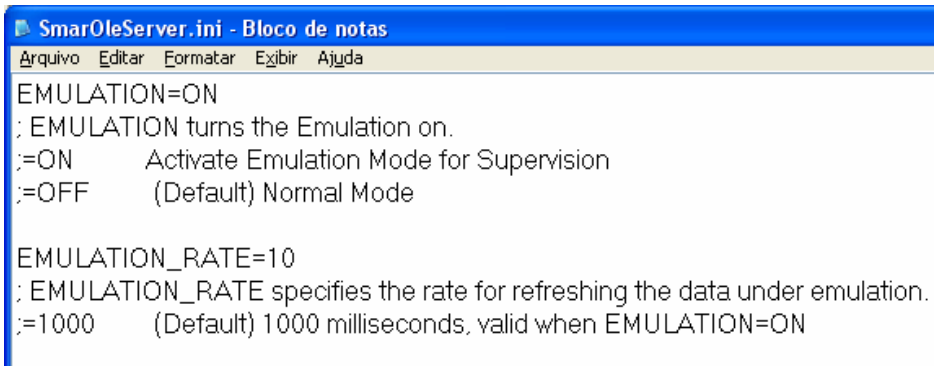

Figura 105 - Arquivo de configuração do Servidor OPC

A configuração “EMULATION=ON” indica para o Servidor OPC que ele deve emular valores. O valor configurado para o parâmetro “EMULATION_RATE” define o intervalo de tempo em milisegundos para a atualização dos valores gerados.

Com esta configuração, puderam ser gerados 16 eventos *Data Change* por segundo, ou seja, um pulso a cada 62,5 milisegundos. Neste caso também foram efetuadas quatro tomadas de monitoramento com tempo de 15 minutos e nas quatro amostragens os valores não se alteraram. A Tabela 08 demonstra a média dos valores obtidos.

Tabela 08 - Tempo Médio de Transmissão no Melhor Caso

Período de Amostragem, em segundos	**Quantidade de *Data Change* Gerados**	**Quantidade de *Data Change* Enviados**	**Tempo médio de envio de um *Data Change*, em segundos**
900	14393	14393	0,06

A Tabela 09 exibe uma amostragem dos valores de tempo obtidos na transmissão entre o Módulo Servidor e o Módulo Cliente.

Tabela 09 - Tempo de Transmissão no Melhor Caso.

***Data Change* Transmitido**	**Servidor**	**Cliente**
1	17:25:59	17:25:59
2	17:25:59	17:25:59
3	17:25:59	17:25:59
4	17:25:59	17:25:59
5	17:25:59	17:25:59
6	17:25:59	17:25:59
7	17:25:59	17:25:59
8	17:25:59	17:25:59
9	17:25:59	17:25:59
10	17:25:59	17:25:59
11	17:25:59	17:25:59
12	17:25:59	17:25:59
13	17:25:59	17:25:59
14	17:25:59	17:25:59
15	17:25:59	17:25:59
16	17:25:59	17:25:59
17	17:26:00	17:26:00
18	17:26:00	17:26:00
19	17:26:00	17:26:00
20	17:26:00	17:26:00
21	17:26:00	17:26:00
22	17:26:00	17:26:00
23	17:26:00	17:26:00
24	17:26:00	17:26:00
25	17:26:00	17:26:00
26	17:26:00	17:26:00
27	17:26:00	17:26:00
28	17:26:00	17:26:00
29	17:26:00	17:26:00
30	17:26:00	17:26:00
31	17:26:00	17:26:00
32	17:26:00	17:26:00
33	17:26:01	17:26:01
34	17:26:01	17:26:01

A Figura 106 exibe o gráfico relativo a essa diferença de tempo.

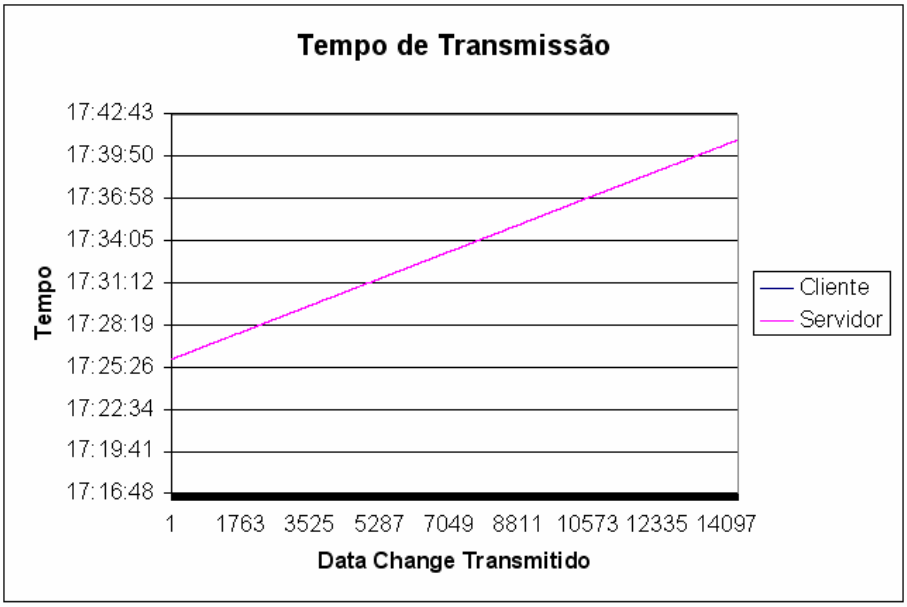

Figura 106 - Gráfico de tempo de transmissão do melhor caso

Os testes demonstraram que o desempenho do protótipo está diretamente relacionado com as taxas de transmissão disponíveis. No caso crítico, que utilizou uma linha discada a uma taxa média de 36 kbps, o desempenho cai drasticamente em relação ao melhor caso, onde uma rede Ethernet com taxa de 100 Mbps foi utilizada.

Observa-se que a velocidade de transmissão do valor de um *Data Change* a cada 0,062s pode ser considerada uma boa taxa em casos de monitoramento de dados em uma rede de automação. Conclui-se então que, em condições ideais de velocidade de transmissão, o protótipo oferece um bom desempenho na transmissão de dados.

4.3 ESTUDO DE APLICAÇÃO

Este seção trata da aplicação prática do aplicativo *EventoWebService*. A Smar Equipamentos Industriais Ltda., uma das líderes mundiais em tecnologia de automação industrial, desenvolve e comercializa equipamentos de campo e softwares de

configuração e supervisão para a área industrial. Entre os softwares desenvolvidos pela Smar está o *System302*.

O *System302* é um sistema completo de Automação Industrial e Empresarial abrangendo os protocolos Foundation™ Fieldbus, HART, Profibus e DeviceNet, e entre os aplicativos disponíveis neste pacote está o Gerenciador de Ativos chamado *AssetView* e as plataformas para Operação e Supervisão de Processos. O foco deste estudo de caso é a ferramenta gerencial de equipamentos de campo, via Internet - o *AssetView*.

O *AssetView* executa a configuração e calibração de equipamentos, monitora a condição da planta, faz o diagnóstico de equipamentos e armazena suas informações. Para efetuar a calibração e configuração de um determinado equipamento, da Smar ou de outros fabricantes, o *Assetview* disponibiliza uma página de Internet específica para o equipamento em questão.

Uma dificuldade no uso desta ferramenta é o desenvolvimento das páginas para equipamentos de outros fabricantes. Quando um cliente adquire o *AssetView* e possui um equipamento que não esteja dentro da vasta lista de equipamentos atendidas pelo *AssetView*, uma página específica para este equipamento deve ser desenvolvida. Como o equipamento é de outro fabricante, é necessário que o cliente envie este equipamento para a Smar para que a Equipe de Suporte analise os parâmetros necessários para desenvolver a página que permite a calibração e configuração do equipamento. Depois que a página é desenvolvida, uma série de teste de configuração e calibração do equipamento é feita.

4.3.1 ANÁLISE DO RESULTADO

Utilizando os componentes de comunicação implementados no protótipo apresentado neste trabalho, não será mais necessário enviar os equipamentos para a Smar. O cliente terá apenas que instalar o Módulo Servidor do *EventoWebService* e enviar para a Smar as informações sobre o Servidor Web e o Servidor OPC, para que se possa efetuar a comunicação entre o Módulo Cliente e o Módulo Servidor. Isto permitirá que a Equipe de Suporte da Smar execute uma série de testes no equipamento para levantar suas configurações e parâmetros e criar a página do equipamento. Estando pronta a página, a Equipe de Suporte utiliza os componentes de comunicação desenvolvidos para o protótipo *EventoWebService*, executa todos os testes necessários e libera a página para uso.

5 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi analisar a viabilidade de sistemas baseados em aplicações distribuídas e propor uma solução para a aquisição de dados distribuídos no setor de automação industrial. A principal motivação foi o interesse crescente da indústria em monitorar equipamentos muitas vezes instalados em espaços físicos distantes, porém interligados pela grande rede mundial de computadores, a Internet.

Várias tecnologias, fundamentais para o desenvolvimento de aplicativos distribuídos e executados sobre a Internet, foram apresentadas e utilizadas no desenvolvimento deste estudo, tais como UML, Microsoft .NET, ASP.NET, *Web Services* e Serviços de Componentes. Entre as várias tecnologias disponíveis para a Internet, a que melhor atendeu as necessidades apresentadas foi o conjunto de tecnologias conhecidas com *Web Services*.

Os *Web Services* ofereceram mecanismos eficientes para a comunicação entre processos através da Internet utilizando protocolos conhecidos, como o protocolo HTTP, para ultrapassar as barreiras comumente impostas por administradores de redes. Utilizando esses mecanismos juntamente com a plataforma de desenvolvimento de software Microsoft .NET, foi desenvolvido um protótipo de monitoramento à distância composto por dois módulos: Módulo Cliente e Módulo Servidor. A configuração necessária para o monitoramento à distância e o resultado deste monitoramento foram armazenados em um banco de dados.

Para o teste do protótipo, foi utilizado um kit didático com equipamentos baseados no protocolo de comunicação Foundation™ Fieldbus. A comunicação dos dados da planta industrial com o módulo servidor foi disponibilizada através de um Servidor OPC, ou seja, através de uma tecnologia padronizada e amplamente utilizada na indústria. Os *Web Services* implementaram a troca de dados entre o Módulo Servidor e o Módulo Cliente, e um sistema de eventos gerenciou a comunicação entre os *Web Services* e os aplicativos.

5.1 CONCLUSÃO

Com relação aos resultados, verificou-se que o protótipo atendeu a todos os itens propostos na sua análise de requisitos, podendo-se considerar que o protótipo foi validado. No caso do desempenho, foi constatado que ele está relacionado diretamente com a velocidade disponível para a transmissão dos dados.

Ainda, pode-se constatar que este trabalho satisfez várias das atuais exigências da indústria referentes à aquisição de dados, tais como:

- Portabilidade: pelo uso de tecnologias como Servidor OPC, XML;
- Interoperabilidade: a utilização de XML, que é um formato texto, permite a interoperabilidade com outros sistemas;
- Robustez e Escalabilidade: ligadas diretamente ao meio de transmissão.

5.2 SUGESTÕES DE TRABALHOS FUTUROS

Utilizando o protótipo proposto neste trabalho, é possível identificar novas funcionalidades e serviços de aquisição remota, tais como:

- Serviços para obter automaticamente a topologia, ou seja, a relação dos equipamentos instalados na rede industrial, onde o módulo servidor esteja instalado, e alimentar a base de dados do módulo cliente sem que o usuário tenha que configurar toda a informação;
- Serviços para obter automaticamente todas as tags utilizadas por um determinado equipamento e armazená-las na base de dados do módulo cliente, sem que o usuário tenha que configurar toda a informação.
- Implementar sistema de segurança para evitar que usuários ou sistemas não autorizados possam acessar ou interagir com o sistema.

REFERÊNCIAS

ARQUITETURA foundation fieldbus. (2008). Sertãozinho: SMAR Equipamentos Industriais Ltda. Disponível em:<http://www.smar.com/brasil2/fieldbus.asp>. Acesso em: 20 mar. 2008.

BUILDING COM+ applications using microsoft .NET enterprise services. (2002). Washington: Microsoft. Apostila.

CHAU, T.D.; KHAI, N.N. (2007). Web-based data monitoring and supervisory control. In: INTERNATIONAL SYMPOSIUM ON ELECTRICAL & ELECTRONICS ENGINEERING, 2007, Ho Chi Minh City. **Proceedings…** Ho Chi Minh City: University of Technology; Faculty of Electrical and Electronics Engineering. p.

CUNHA, D. (2002). **Web services, SOAP e aplicações web**. Disponível em:<http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html>. Acesso em: 10 mar. 2008.

EIDE, A. et al. (2001). **Professional ASP.NET web services**. Chicago: Wrox.

IWANITS, F.; LANGE, J. (2002). **OPC fundamentals, implementation, and application**. 2nd ed. Scottsdale: Panoramix Services LLC.

KAPSALIS, V. et al. (2002). Web gateway: a platform for industry services over internet. In: IEEE INTERNATIONAL SYMPOSIUM INDUSTRIAL ELECTRONICS, 2002, New York. **Proceedings…** New York: IEEE. v.1, p.73-77.

_____. (2003). Architecture for web-based services integration. In: ANNUAL CONFERENCE OF INDUSTRIAL ELECTRONICS SOCIETY, 29., 2003, New York. **Proceedings…** New York: IEEE. v.1, p.866-871.

KIRTLAND, M. (1998). **Designing component-based applications**. Washington: Microsoft. (Microsoft Programming Series).

KONDOR, R. (2008). **OPC tutorial**. Disponível em:<http://www.matrikonopc.com/resources/opc-tutorials.aspx>. Acesso em: 27 fev. 2008.

LÖWY, J. (2001). **COM & .NET component services**. Cambridge: O'Reilly Media.

MARINO, A.F. (2000). **A Internet na automação, instrumentação e controle de processos**. Disponível em:<http://www.engecomp.com.br/marino.htm>. Acesso em: 15 fev. 2008.

MATA, R.S. (2005). Descobrindo a tecnologia foundation fieldbus - parte 1. **Revista Mecatrônica Atual**, São Paulo, n.24, p.53-59, out.

PAGNANO, M.A.O. (2003). AssetView a ferramenta de gerenciamento on-line de ativos. **C & I**: controle & instrumentação, São Paulo, n.80, p.50-55, maio.

PERES FILHO, G.F.P.; MATA, R.S. (2004). Tecnologia foundation fieldbus. **Revista Mecatrônica Atual**, São Paulo, n.18, p.54-59, out.

RECKZIEGEL, M. (2006). **Protocolo de transporte padrão**: SOAP. Disponível em:<http://imasters.uol.com.br/artigo/4379/webservices/protocolo_de_transporte_padrao_-_soap/>. Acesso em: 5 mar. 2008.

REHBEIN, D.A. (2008). **OLE for process control (OPC) primer**. Disponível em:<http://www.hartcomm.org/download/opcprimer.pdf>. Acesso em: 1 mar. 2008.

SHIMANUKI, Y. (1999). Ole for process control (OPC) for new industrial automation systems. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, 1999, Tokio. **Proceedings...** New York: IEEE. v.6, p.1048-1050.

SOUZA, A.J.; OLIVEIRA, L.C. (2006). Princípios da automação industrial: estudo, processos, arquiteturas e evoluções. **Cadware Indústria**, São Paulo, v.3, n.3, ago. Disponível em:<http://www.cwbookstore.com.br/cet_ta.cfm>. Acesso em: 10 nov. 2007.

SOUZA, A.J. et al. (2006). Supervisão de processos industriais usando web service. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 12., 2006, Salvador. **Anais...** Salvador: SBA. 1 CD ROM.

STENERSON, J. (2002). **Industrial automation and process control**. Upper Saddle River: Prentice Hall.

TAN, V.V. et al. (2007). A Novel framework for building distributed: data acquisition and monitoring systems. **Journal of Software**, Ulsan, v.2, n.4, p.70-79, Oct.

WARNER, S.; KAGHAZCHI, H. (2007). Development of web-based software for a multi-fieldbus diagnosis tool. In: IEEE CONFERENCE ON PUBLICATION EMERGING TECHNOLOGIES & FACTORY AUTOMATION, 2007, New York. **Proceedings…** New York: IEEE. p.720-723.

APÊNDICE A

SISTEMA MODULAR DFI302

O DFI302 *(Fieldbus Universal Bridge)* é um sistema modular, microprocessado, com alta capacidade de expansão (Figura 107). Ele possui quatro canais Foundation™ Fieldbus H1 e interface para a rede *High Speed Ethernet* (HSE - 100 Mbps). O DFI302 comunica-se com o micro-computador através de uma placa de rede, possibilitando monitoração e atuação através do OPC. Outra característica do DFI302 é a integração com equipamentos com protocolo Foundation™ Fieldbus, e equipamentos com os mais variados tipos de conexões. Embora haja uma grande variedade de instrumentos de campo inteligentes no mercado que utilizam o protocolo de comunicação Foundation™ Fieldbus, muitas aplicações requerem conexões com equipamentos e dispositivos de outros protocolos que necessitam interagir com a rede de comunicação. Exemplos desses equipamentos são os instrumentos convencionais do tipo transmissores de sinal em corrente ou tensão provenientes de termosensores, ou de sinais discretos de entrada e saída.

Figura 107 - DFI302 da Smar

A Tabela 10 mostra as características dos principais módulos do DFI302.

Tabela 10 - Principais Características do DFI302.

Controladores	DF51	Controlador com 1 porta 10 Mbps e 4 canais H1 Foundation™ Fieldbus
	DF62	Controlador com 1 porta 100 Mbps e 4 canais H1 Foundation™ Fieldbus
	DF63	Controlador com 2 portas 10 Mbps e 4 canais H1 Foundation™ Fieldbus
	DF65	Co-processador Lógico
	DF73	Controlador HSE/Profibus-DP com 2 portas Ethernet 100 Mbps e 1 canal Profibus
	DF75	Controlador HSE
Entrada DC	DF11	2 Grupos de 8 entradas 24VDC (Isoladas)
	DF12	2 Grupos de 8 entradas 48VDC (Isoladas)
	DF13	2 Grupos de 8 entradas 60VDC (Isoladas)
	DF14	2 Grupos de 8 entradas 125VDC (Isoladas)
	DF15	2 Grupos de 8 entradas 24VDC (Sink) (Isoladas)
Entrada AC	DF16	2 Grupos de 4 entradas 120VAC (Isoladas)
	DF17	2 Grupos de 4 entradas 240VAC (Isoladas)
	DF18	2 Grupos de 8 entradas 120VAC (Isoladas)
	DF19	2 Grupos de 8 entradas 240VAC (Isoladas)
Chaves	DF20	1 Grupo de 8 chaves ON/OFF
Saída DC	DF21	1 Grupo de 16 saídas Coletor Aberto
	DF22	2 Grupos de 8 saídas Transistor (Source) (Isoladas)
Saída AC	DF23	2 Grupos isolados de 4 saídas 120/240VAC
	DF24	2 Grupos isolados de 8 saídas120/240VAC
Saída AC/DC	DF25	2 Grupos de 4 saídas Relé NA
	DF26	2 Grupos de 4 saídas Relé NF
	DF27	1 Grupo de 4 saídas Relé NA e 1 Grupo de 4 saídas Relé NF
	DF28	2 Grupos de 8 saídas Relé NA
	DF29	2 Grupos de 4 saídas Relé NA (sem proteção RC)
	DF30	2 Grupos de 4 saídas Relé NF (sem proteção RC)
	DF31	1 Grupo de 4 saídas NA e 4 saídas NF (sem proteção RC)
	DF69	2 Grupos de 8 saídas NA (com proteção RC)
	DF71	2 Grupos de 4 saídas Relé NA (sem proteção RC) – Max 10 mA

	DF72	2 Grupos de 4 saídas Relé NF (sem proteção RC) – Max 10 mA
Entrada Analógica	DF44	1 Grupo de 8 entradas analógicas Tensão-Corrente com resistores internos Shunt
	DF45	1 Grupo de 8 entradas analógicas de sinais de baixo nível para TC, RTD, mV e Ohm
	DF57	1 Grupo de 8 entradas analógicas diferenciais com resistores internos Shunt
Saída Analógica	DF46	1 Grupo de 4 saídas analógicas de Tensão/Corrente
Entrada DC e Saída AC/DC	DF32	1 Grupo de 8 entradas 24VDC e 1 Grupo de 4 Relés NA (Isolados)
	DF33	1 Grupo de 8 entradas 48VDC e 1 Grupo de 4 Relés NA (Isolados)
	DF34	1 Grupo de 8 entradas 60VDC e 1 Grupo de 4 Relés NA (Isolados)
	DF35	1 Grupo de 8 entradas 24VDC e 1 Grupo de 4 Relés NF (Isoladas)
	DF36	1 Grupo de 8 entradas 48VDC e 1 Grupo de 4 Relés NF (Isolados)
	DF37	1 Grupo de 8 entradas 60VDC e 1 Grupo de 4 Relés NF (Isoladas)
	DF38	1 Grupo de 8 entradas 24VDC e 1 Grupo de 2 Relés NA e 2 Relés NF (Isolados)
	DF39	1 Grupo de 8 entradas 48VDC e 1 Grupo de 2 Relés NA e 2 Relés NF (Isolados)
	DF40	1 Grupo de 8 entradas 60VDC e 1 Grupo de 2 Relés NA e 2 Relés NF(Isolados)
Entrada de Pulso	DF41	2 Grupos de 2 entradas de pulso 24VDC de baixa freqüência (0 - 100Hz)
	DF42	2 Grupos de 8 entradas de pulso 24VDC de alta freqüência (0 - 10KHz)
	DF67	2 Grupos de 8 entradas de pulso AC de alta freqüência (0 - 10 KHz)
Interfaces	DF58	Interface EIA 232 / 485
	DF61	Switch Ethernet 10/100 Mbps
Fonte de Alimentação	DF50	Fonte de Alimentação AC para IMB com capacidade de redundância
	DF52	Fonte de Alimentação AC para Fieldbus
	DF56	Fonte de Alimentação DC para IMB com capacidade de redundância
	DF60	Fonte de Alimentação DC para Fieldbus
Impedância para Fonte de Alimentação	DF49	Impedância para Fonte de Alimentação Fieldbus - 2 portas
	DF53	Impedância para Fonte de Alimentação Fieldbus - 4 portas
Barreira de Segurança Intrínseca	DF47	Barreira de segurança intrínseca para Fieldbus

Fonte: SMAR

SYSCON

O *Syscon* é uma ferramenta que além de configurar os dispositivos de campo, permite criar e transferir a estratégia de controle elaborada pelo usuário para os

instrumentos. A estratégia de controle também pode ser editada utilizando-se um dos modelos de estratégias de controle com valores padrão, disponíveis na biblioteca de modelos do *Syscon*.

O *Syscon* verifica cada estratégia de controle definida, procurando por inconsistências e indicando parâmetros que devem ser corrigidos antes que a nova configuração comece a comunicar com a planta. A ferramenta de configuração também gera relatórios completos sobre cada configuração, incluindo lista de instrumentos, diagramas de malha de controle e diagramas de blocos funcionais.

A configuração dos instrumentos consiste basicamente na interconexão lógica dos diversos blocos funcionais implementados em cada equipamento da rede, e na definição dos parâmetros de controle de cada bloco. Os esquemas internos dos blocos funcionais podem ser visualizados quando o projeto de controle está sendo configurado no *Syscon*. Esta característica facilita o entendimento interno do bloco, facilitando a configuração dos parâmetros internos bem com a interligação das entradas e saídas dos blocos funcionais.

A configuração do sistema Foundation™ Fieldbus no *Syscon* pode ser feita em modo off-line ou no modo on-line. A Figura 108 mostra uma tela típica do *Syscon*. Na parte da esquerda é configurada a estratégia de controle interligando os blocos adicionados ao projeto. Na parte da direita são mostrados os quatro canais da rede, sendo que cada canal é configurado independentemente, assim como os equipamentos conectados a rede de comunicação.

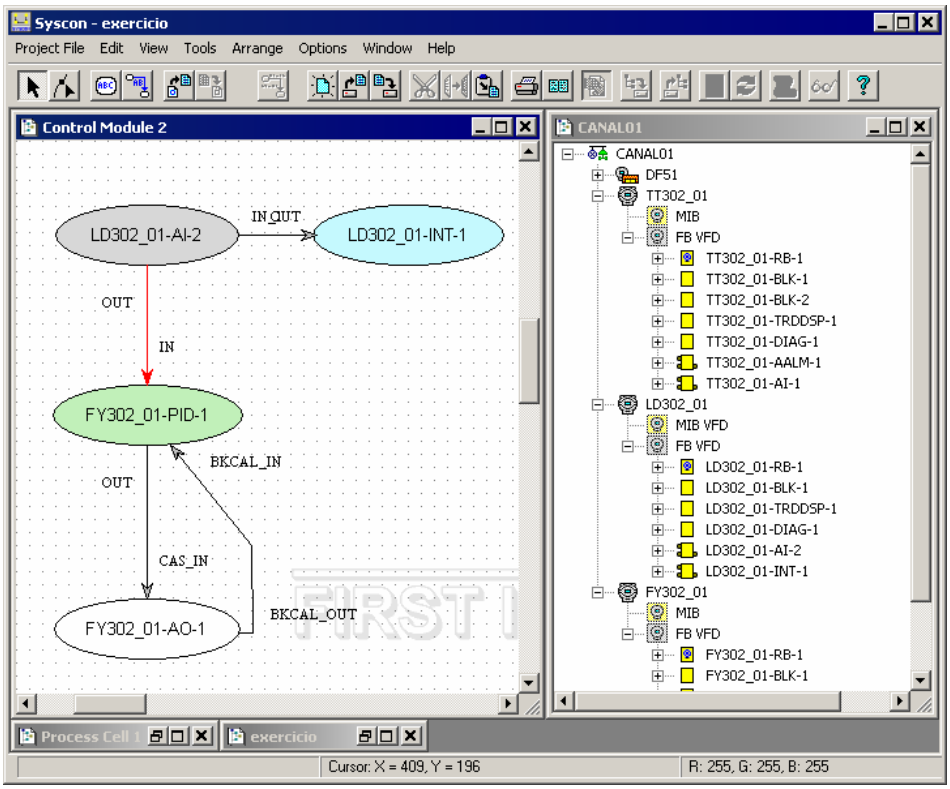

Figura 108 - Tela Típica do *Syscon*

Os blocos trocam informações através de uma entidade chamada link. O link tem a função de conectar um parâmetro de saída de um bloco funcional a um ou mais parâmetros de entrada de outros blocos funcionais. Se os blocos ligados estiverem localizados no mesmo equipamento, diz-se que essa ligação é um link interno. Se a conexão for entre blocos de equipamentos distintos, tem-se um link externo, que consome banda do segmento H1 durante a troca de dados.