Paulo Vitor Lima Souza

Conexão de um dispositivo Ethernet a um sistema supervisório usando o padrão OPC

Feira de Santana – BA

Agosto / 2008

Paulo Vitor Lima Souza

Conexão de um dispositivo Ethernet a um sistema supervisório usando o padrão OPC

Monografia apresentada à Banca de Graduação em Engenharia de Computação da Universidade Estadual de Feira de Santana referente ao Trabalho de Conclusão de Curso para obtenção do título de Bacharel em Engenharia de Computação.

Orientador:
Prof. Dr. Anfranserai Morais Dias

Curso de Engenharia de Computação
Departamento de Tecnologia
Universidade Estadual de Feira de Santana

Feira de Santana – BA

Agosto / 2008

Monografia de Projeto de Final de Graduação sob o título "Conexão de um dispositivo Ethernet a um sistema supervisório usando o padrão OPC"apresentada em 07 de agosto de 2008 à banca examinadora constituída pelos professores:

Prof. Dr. Paulo Cesar Machado de Abreu Farias
Departamento de Ciências Exatas - UEFS
Examinador

Prof. Msc. Vitor Leão Filardi
Departamento de Tecnologia - UEFS
Examinador

Prof. Dr. Anfranserai Morais Dias
Departamento de Tecnologia - UEFS
Orientador

Dedico esse trabalho à minha mãe,
uma pessoa verdadeira, lutadora, incansável, trabalhadora,
detentora de características que marcam qualquer filho,
ao meu saudoso pai, pelo exemplo de integridade,
de pessoa correta, de uma humildade sem igual,
que me ensinou a ter tantos dos predicados que ele possuía
e valores que só se adquirem com uma boa educação,
aos meus irmãos, pelo companheirismo incondicional,
aos meus amigos pelos bons momentos e histórias para contar,
e à minha namorada, cuja simples presença já me faz sorrir.

Agradecimentos

Ao professor Doutor Anfranserai Morais Dias, pela orientação e auxílio na elaboração deste trabalho;

Ao professor Doutor Gustavo Henrique Machado de Arruda, pelo incentivo e motivação, bem como sua disposição de estar presente na solução de dificuldades;

Aos colegas da Graduação em Engenharia de Computação da UEFS.

"A atividade da engenharia, enquanto permanecer atividade,
pode levar a criatividade do homem a seu grau máximo;
mas, assim que o construtor pára de construir e se entrincheira
nas coisas que fez, as energias criativas se congelam,
e o palácio se transforma em tumba."

Marshall Berman

Resumo

Este trabalho trata do desenvolvimento de um sistema que permite a conexão de um dispositivo embarcado com suporte à comunicação via Ethernet, plataforma de desenvolvimento TINI (Tiny InterNet Interface), a um sistema supervisório utilizando um padrão baseado na tecnologia OLE (Object Linking and Embedding)da Microsoft Corporation, o padrão OPC (OLE for Process Control). Basicamente, o padrão OPC estabelece regras, interfaces padrões, sem que os clientes das aplicações precisem tratar os diferentes formatos e protocolos de comunicação existentes. Cabe aos clientes tratar os dados de acordo às interfaces oferecidas, acessando qualquer dispositivo da mesma maneira. O projeto consiste no desenvolvimento de um servidor OPC baseado na especificação Data Access para o TINI, de forma que alguns dados deste possam ser disponibilizados aos supervisórios (clientes OPC). Serão apresentados, neste trabalho, a implementação do servidor OPC Data Access (tratamento dos dados adquiridos do dispositivo e disponibilização dos dados através das interfaces da especificação), a transmissão de dados do dispositivo para o servidor OPC (implementação da comunicação ponto-a-ponto e gerenciamento dos dados a serem transmitidos no dispositivo) e um protótipo de automação industrial ligando as partes desenvolvidas a um sistema supervisório.

Abstract

This project is the development of a system that allows the connection of an embedded device with support for communication via Ethernet, TINI (Tiny InterNet Interface) development platform, to a supervisory system using an technology based in OLE (Object Linking and Embedding) from Microsoft Corporation, the OPC (OLE for Process Control) standard. Basically, the standard OPC defines rules, standard interfaces, without the client applications need to deal with the different formats and protocols of communication. The clients only treat the data as the interfaces offered, then, they access any device in the same way. The project is the development of a TINI server based on the OPC Data Access specification, with some of these data may be available to supervisory (OPC client). They are then shown, in this study, the implementation of the OPC Data Access server (processing of data acquired from the device and availability of data through the interface's specification), the transmission of data from device to the OPC server (implementation of the communication point-to-point and management of data to be transmitted to the device) and a prototype of industrial automation, linking the developed parts at a supervisory system.

Sumário

Lista de Figuras

Lista de Tabelas

Código-fonte

Lista de Abreviaturas

ANSI - *American National Standards Institute*

API - *Application Programming Interface*

CLP - Controlador Lógico Programável

CLSID - *Class Identifier*

COM - *Component Object Model*

DCE - *Distributed Computing Environment*

DCOM - *Distributed Component Object Model*

DDE - *Dynamic Data Exchange*

DLL - *Dynamic-Link Library*

GUID - *Global Unique Identifier*

IDL - *Interface Definition Language*

IHM - Interface Homem-Máquina

IP - *Internet Protocol*

ISA - *Industry Standard Architecture*

JDK - *Java Development Kit*

MFC - *Microsoft Foundation Class*

MSDN - *Microsoft Developer Network*

MSVC++ - *Microsoft Visual C++*

OLE - *Object Linking and Embedding*

OPC - *OLE for Process Control*

OPC-AE - *OPC Alarms and Events*

OPC-DA - *OPC Data Access*

OPC-DX - *OPC Data eXange Specification*

OPC-HDA - *OPC Historical Data Access*

OPC XML-DA - *OPC XML Data Access Specification*

OSF - *Open Software Foundation*

SCADA - *Supervisory Control And Data Acquisition*

SPI - *Serial Peripheral Interface*

TCP - *Transmission Control Protocol*

TINI - *Tiny InterNet Interface*

XML - *Extensible Markup Language*

1 Introdução

Numa rede industrial há uma crescente preocupação em se minimizar custos gerais, maximizar a qualidade do produto e otimizar a produtividade. A otimização da produtividade tem como fator de desempenho a forma como os diversos dispositivos que compõe a rede industrial comunicam-se entre as diversas camadas da arquitetura de informação num processo de manufatura, ligando a gerência de negócios, gerência de processos e o chão de fábrica.

Os sistemas de automação industrial utilizam tecnologias de computação e comunicação para automatizar a monitoração e controle de processos industriais, efetuando coleta de dados em ambientes complexos, eventualmente dispersos geograficamente, e apresentando-os para o operador com recursos gráficos elaborados e conteúdo multimídia.

Tais sistemas, denominados SCADA (*Supervisory Control And Data Acquisition*), são geralmente usados para monitorar ou controlar processos químicos, elétricos, físicos ou de transporte. A interface homem-máquina (IHM) é a parte do sistema SCADA responsável por apresentar os dados do processo para o operador. Uma IHM/SCADA pode consultar um banco de dados remoto, apresentar gráficos com históricos de variáveis do processo, agendar procedimentos de manutenção, detalhar esquemas de equipamentos, dentre outras tarefas.

Assim, a grande quantidade de esforço e, portanto, de custo envolvido na implementação de sistemas de automação é devido a resolução de problemas relativos à interface entre diferentes tipos de equipamentos de hardware e software (PATTLE; RäMISCH, 1997) (ZHENG; NAKAGAWA, 2002). Pela grande diversidade de dispositivos, tecnologias, e sistemas SCADA, uma das dificuldades existentes numa rede industrial é como integrar estes recursos.

Em geral, cada fabricante de dispositivo utiliza um protocolo de comunicação diferente, dispondo seus dados de forma diferente e não padronizada, variando até mesmo entre as diferentes versões do dispositivo. Para que sistemas de gerenciamento de dis-

positivos possam interpretar estes dados, faz-se necessário a utilização de drivers que especifiquem a forma de reconhecer os dispositivos, que são fornecidos pelos próprios fabricantes.

A utilização de drivers, por sua vez, não resolve o problema de integração entre dispositivos e sistemas de gerenciamento de atividade e processos, pois, por vezes estes causam incompatibilidade entre si, principalmente se dependerem dos mesmos recursos.

Outro fator levado em consideração quanto a integração de dispositivos utilizando drivers é que ainda assim os desenvolvedores de IHM e outras ferramentas de gerência devem tratar os protocolos e interfaces de comunicação. Desta forma, lidam com as diversas formas que os drivers permitem se conectar ao dispositivo e tratar os dados dispostos.

O desenvolvimento de drivers e o modo de acesso a estes para cada dispositivo da arquitetura industrial é ineficiente, consome tempo de desenvolvimento, provoca riscos adicionais ao sucesso e completude de projetos, além de não garantir a integração dos diversos dispositivos (PATTLE; RäMISCH, 1997).

A chave para a realização dessa integração é uma arquitetura aberta e efetiva de comunicação baseada no acesso a dados e não no tipo de dados (PATTLE; RäMISCH, 1997) (ZHENG; NAKAGAWA, 2002). Em maio de 1995, cinco empresas em cooperação com a Microsoft Corporation criaram um comitê com o objetivo de remover estas barreiras de comunicação.

O objetivo do comitê era prover uma interface padrão para troca de dados baseado no acesso a estes, que permita simplificar o processo de desenvolvimento de drivers de entrada e saída e melhorar a performance dos sistemas. Como resultado, foi criado a especificação técnica OPC (*OLE for Process Control*) baseada nas tecnologias OLE/COM (*Object Linking and Embedding / Component Object Model*) da Microsoft.

A especificação OPC torna possível a troca de dados e interoperabilidade entre sistemas/dispositivos, sistemas de automação/controle e softwares de gerenciamento de uma fábrica. Atualmente a especificação OPC encontra-se em sua versão 3.0.

O sucesso do OPC vem em grande parte por causa da tecnologia COM, desenvolvida em 1993. COM é usada para permitir a comunicação entre processos e a criação dinâmica de objetos em qualquer linguagem de programação que suporte a tecnologia.

Essa tecnologia estabelece um padrão binário, possibilitando o desenvolvimento de aplicativos baseado na interface de programação de aplicativos (*Application Programming*

Interface - API) do Windows (IWANITZ; LANGE, 2005) (MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION, 1995). Permite também a reutilização de objetos sem o conhecimento de sua implementação interna, pois força o desenvolvedor a fornecer uma interface bem definida, independente da linguagem de programação a implementá-la.

Por outro lado, as redes de computadores com padrão Ethernet são cada vez mais aceitas na área industrial, havendo uma grande tendência em migrar outras soluções existentes como RS232, RS485 e CAN (WONDERWARE CORPORATION, 2001).

Embora este padrão seja normalmente associado ao protocolo TCP/IP (Transmission Control Protocol/Internet Protocol), nas redes industriais existem várias outras opções de protocolo. Estas opções vão desde padrões abertos, até protocolos proprietários, porém a prática tem sido aproveitar protocolos existentes e construir a camada de aplicação para TCP/IP.

Este projeto visa o desenvolvimento de um servidor OPC que possibilite a conexão de sistemas SCADA com um sistema embarcado baseado na plataforma de desenvolvimento TINI (Tiny InterNet Interface). O TINI é composto por um microcontrolador da Dallas Semiconductor e um ambiente de programação em Java, e fornece um canal de comunicação de rede, permitindo o estabelecimento de protocolos que conectem dispositivos na rede Ethernet (LOOMIS, 2001).

O estabelecimento de um servidor OPC para a plataforma TINI possibilita integrar um dispositivo Ethernet em um ambiente de automação industrial, sem que seja apenas para soluções específicas e dedicadas. Assim, pretende-se, com este projeto, criar exemplos de automação industrial, utilizando a linguagem C++ para desenvolvimento, seguindo padrões bem estabelecidos para troca de dados entre processos, usando ferramentas típicas para desenvolvimento de sistemas SCADA e testando sua interação com estes.

Para fazer as discussões sobre o conteúdo deste trabalho, ele é dividido da seguinte forma:

No capítulo 2, os fundamentos sobre OPC são apresentados, comparando o padrão com as tecnologias existentes para troca de dados em uma rede industrial. Mostra-se a hierarquia da informação em um processo industrial e a arquitetura das aplicações OPC. O capítulo termina com a discussão da especificação destinada ao acesso aos dados de um servidor OPC, implementado para a plataforma TINI.

O capítulo 3 trata sobre o desenvolvimento do projeto, com a construção do servidor, a comunicação entre o servidor e o dispositivo, e a disponiblização dos dados pelo TINI.

São ilustrados os diagramas de blocos do sistema construído e o procedimento para a construção do servidor.

Os resultados obtidos com as implentação do sistema, os testes de validação com os sistemas supervisórios e alguns problemas encontrados para a construção do sistema são apresentados no capítulo 4, enquanto as considerações finais e perspectivas para futuros trabalhos são explanados no capítulo 5.

2 OLE for Process Control - OPC

Uma das dificuldades na implementação de um sistema industrial é a integração dos diferentes componentes dos níveis da arquitetura da informação em um processo de manufatura. Os dados de distintos sistemas possuem diferentes formatos e protocolos de comunicação.

Um mecanismo que possibilite esta integração é muito importante, uma vez que dispositivos de chão-de-fábrica são conectados a um sistema SCADA. Desenvolvedores de software para monitoramento de processos, controle e sistemas de gerencia de dados, necessitam que seja fornecido um driver individual para cada dispositivo (PATTLE; RäMISCH, 1997). Estes drivers permitem a conexão dos sistemas SCADA aos diferentes dispositivos de entrada e saída.

Através do mecanismo de drivers, os sistemas SCADA podem acessar estes dispositivos, porém é necessário ainda tratamento dos diferentes formatos de dados e protocolos de comunicação. Além disso, diferentes drivers destes componentes podem causar conflitos entre si, e mudanças nas capacidades do hardware, como a forma de comunicação, podem causar falhas de funcionalidade de alguns deles.

Assim, devido a grande quantidade de dispositivos existentes no mercado e o crescente número de protocolos de comunicação, os desenvolvedores de software viam-se obrigados a desenvolver, manter e gerenciar centenas de drivers (IWANITZ; LANGE, 2005). Uma parte considerável de recursos destas empresas deviam ser destinados ao desenvolvimento e manutenção destes drivers.

Em maio 1995, as companhias Fisher-Rosemont, Intellution, Intuitive Technology, Opto22, Rockwell e Siemens AG, decidiram trabalhar em cooperação para propor uma solução para este problema e formaram a Força Tarefa OPC (OPC *Task Force*). Membros da equipe Microsoft também estavam envolvidos, fornecendo assistência técnica.

A Força Tarefa OPC desenvolveu um padrão de acesso a dados em tempo real sob o sistema operacional Windows, baseado na tecnologia OLE/DCOM (*Distributed Component*

Object Model) da Microsoft. Em agosto de 1996, como resultado, a Especificação OPC Versão 1.0 foi liberada (PATTLE; RäMISCH, 1997) (IWANITZ; LANGE, 2005). Em setembro de 1996, a OPC Foundation foi criada, para coordenar e organizar as especificações.

A OPC Foundation é responsável por atender as necessidades da indústria e adicionar novas funcionalidades ao padrão OPC ou estender as especificações existentes. A estratégia adotada foi a criação de extensões à especificação existente, definição da adição de novas especificações e a realização de modificações para manter a compatibilidade máxima com as versões existentes (IWANITZ; LANGE, 2005).

A importância crescente do padrão OPC, o uso dos produtos baseados nele nas mais diversas áreas, e os novos requisitos tecnológicos, como a independência de uma plataforma para funcionamento e a capacidade de conexão à Internet, criou pontos de contato com outras organizações.

Quando imerge em novas áreas, a OPC Foundation não tem o objetivo de promover padrões existentes de organizações ou reinventar atividades e especificações. O objetivo é combinar as forças de organizações, na qual o OPC determina como os dados são transportados enquanto as outras organizações lidam com o que é transportado, ou seja, quais são os dados e em que tipos de dados são baseados.

Assim, a grande motivação para se criar o padrão OPC foi a necessidade de se estabelecer um mecanismo padrão de comunicação entre diferentes fontes de dados em um processo industrial, sejam eles provenientes de equipamentos de campo ou até mesmo de outros arquivos de dados (ZHENG; NAKAGAWA, 2002) (IWANITZ; LANGE, 2005).

Basicamente, o padrão OPC estabelece as regras para que sejam desenvolvidos sistemas com interfaces comuns para comunicação dos dispositivos de campo (CLP, sensores, etc.) com sistemas de monitoração, supervisão e gerenciamento (PATTLE; RäMISCH, 1997).

A hierarquia da informação em um processo industrial envolve três níveis como observado na Figura 1. Esse padrão obedece a terminologia adotada pela ANSI/ISA Standard 95 (*American National Standards Institute / Industry Standard Architecture*). ISA-95, como é mais conhecido, é um padrão internacional para o desenvolvimento de uma interface entre a gerência do negócio e sistemas de controle.

Os objetivos da ISA-95 são prover uma terminologia consistente para a comunicação, modelos de informação consistentes, e modelos de operação consistentes, que são uma base para esclarecer a funcionalidade das aplicações e como a informação pode ser usada (ISA-95, 1995).

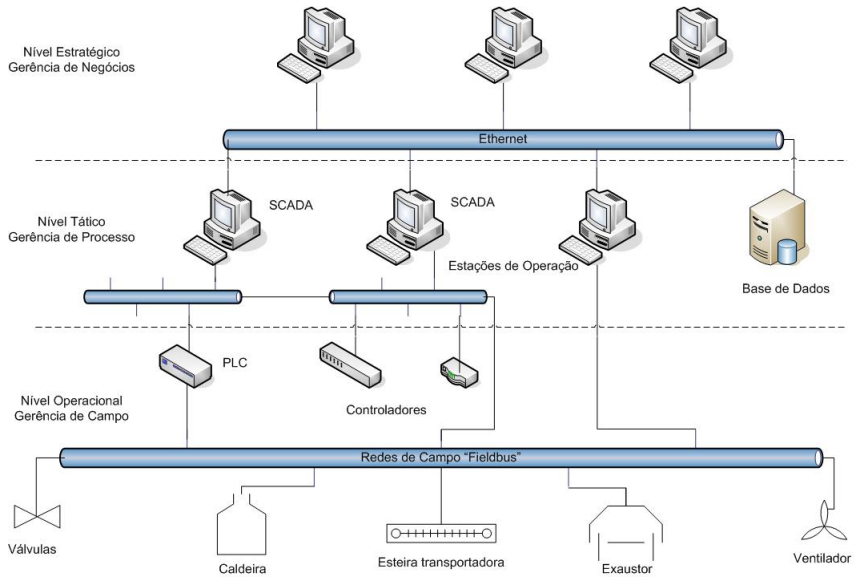

Figura 1: Hierarquia da informação em um processo industrial.

O nível de Gerência de Campo representa o nível mais baixo, onde situam-se os equipamentos de produção de uma fábrica, a aquisição de dados (sensores e atuadores), e unidades de controle (regulagem, monitoramento e proteção), havendo uma exigência maior de tempo de resposta. Com o advento dos equipamentos de campo inteligentes, uma grande variedade de dados provenientes desses equipamentos, como dados de configuração e controle, podem ser disponibilizados para usuários ou mesmo para outras aplicações.

O nível de Gerência de Processo representa o nível intermediário, sendo responsável pela supervisão da produção e dos equipamentos, otimização e execução de operações, visualização da planta da fábrica, armazenamento de dados do processo, geração de registro e histórico de operações. A utilização de sistemas de controle do tipo SCADA permite o controle descentralizado de processos industriais. Os dados fornecidos podem ser considerados conjuntamente de modo a permitir uma gerência efetiva e integrada de todo o processo industrial.

O nível de Gerência do Negócio representa o nível mais alto, responsável pela integração das informações de chão-de-fábrica e os dados corporativos, administrativos e de aspectos financeiros. Os dados e informações podem ser utilizados por aplicações cliente de modo a otimizar a gerência e a integração de todo o processo de manufatura (OPC TASK FORCE, 1998).

Para realizar essa integração de modo efetivo, os fabricantes precisam ter acesso aos dados de chão-de-fábrica, dos processos industriais e integrá-los aos seus sistemas corporativos. Devem, também, poder utilizar as ferramentas disponíveis nos sistemas SCADA

para atingir os objetivos da gerência integrada dos processos industriais.

Observando esta estrutura hierárquica nota-se a necessidade da integração destas informações, através de uma arquitetura baseada na disponibilidade de acesso aos dados, que faça a ligação entre os componentes do ambiente industrial e controladores de *hardware* e dispositivos.

Dados de processo do nível de campo podem ser apresentados em uma planilha eletrônica, dados de estado dos dispositivos e dados do nível de produção podem ser armazenados em uma base de dados sem nenhum problema via OPC, podendo ser processados depois em um sistema de planejamento de produção no nível de gerenciamento (PATTLE; RäMISCH, 1997) (OPC TASK FORCE, 1998).

2.1 Tecnologias OLE e COM/DCOM

O padrão OPC, como o próprio nome indica, é uma aplicação de uma tecnologia OLE, que surgiu em meados de 1990. Este termo descreve a tecnologia Microsoft que realiza sistemas orientados a objetos. Inicialmente, OLE era apenas um protocolo voltado à elaboração de documentos compostos. Essa versão original foi construída sobre os mecanismos de DDE (Dynamic Data Exchange).

DDE descreve um mecanismo de troca de dados entre aplicações dentro da plataforma Windows (PATTLE; RäMISCH, 1997) (IWANITZ; LANGE, 2005). Isto significou, por exemplo, que uma tabela do Excel podia ser copiada em um documento do Word, porém a conexão não era dinâmica, as mudanças não eram transferidas de um documento para outro de forma automática. O principal inconveniente desta solução era a sua baixa largura de banda, o que não é muito apropriado para sistemas de tempo real.

Para atender a este requisito, a Microsoft aprimorou o padrão OLE de forma que a tecnologia permite estabelecer conexões entre objetos, assim, é possível que mudanças numa tabela no Excel sejam transferidas automaticamente para um documento do Word (*linking*) e editado diretamente nele (*embedding*).

O aprimoramento desta tecnologia, OLE2, dirigiu-se a um passo adiante, para o modelo de *softwares* baseados em componentes, assim esta interação poderia ser feita não somente para programas de processamento de texto, deste que a aplicação estivesse suscetível a este modelo. Esta tecnologia foi denominada de *Component Object Model* (COM) em 1995, representando a tecnologia básica da Microsoft para aplicações orien-

tadas a objetos (IWANITZ; LANGE, 2005).

Disponível desde 1996, DCOM permite a interação entre componentes em computadores diferentes. Esta interação foi desenvolvida apoiada no trabalho da Open Software Foundation (OSF), criada em 1980 com o objetivo de tornar possível a computação distribuída em ambientes heterogêneos. Esta iniciativa da OSF resultou em várias especificações na área de computação distribuída, ou ambiente de computação distribuída (DCE - Distributed Computing Environment).

Um ambiente heterogêneo significa que a aplicação pode ser criada em diferentes linguagens de programação, que pode se executada em diferentes arquiteturas de computadores, e interagir sobre várias redes de comunicação (COULOURIS; DOLLIMORE, 2005).

Um dos resultados do trabalho de especificação da OSF foi a definição dos mecanismos de comunicação entre aplicações em diferentes computadores. Neste contexto, é importante determinar como aplicações diferentes entenderão uma a outra e como podem chamar funções de outras aplicações.

Para isso, é necessário transformar os parâmetros cuja representação depende do processo, para uma representação independente do processo especifico, do sistema operacional e da rede de comunicação. Dois componentes, *proxy* e *stub*, são responsáveis por esta transformação, montando e desmontando mensagens (*marshalling*/*demarshalling*) (IWANITZ; LANGE, 2005).

Um cliente chama métodos através do *proxy* para interagir com o servidor (*marshalling*), não acessando-o diretamente. Esta chamada é passada para o *stub* em tempo de execução, e no *stub*, a mensagem é transformada de forma que o servidor possa entendê-la (*demarshalling*) e o método chamado pode ser executado no servidor. Quando o servidor responde ao cliente, o mesmo processo é realizado na direção oposta (PATTLE; RäMISCH, 1997).

Como uma continuação da tecnologia OLE, o COM/DCOM surgiu junto com o sistema operacional Windows NT e foi logo aceito pela indústria. Basicamente, o DCOM é um conjunto de definições para permitir a implementação de aplicações distribuídas em uma arquitetura cliente-servidor.

Desta forma, um cliente pode acessar diferentes servidores ao mesmo tempo e um servidor pode disponibilizar suas funcionalidades para diferentes clientes ao mesmo tempo. A Tabela 1 mostra uma comparação entre a utilização de DDE, OPC/DCOM e drivers.

Em programação orientada a objetos, os objetos criados por diferentes aplicações, em

Tabela 1: Comparação entre tecnologias de suporte à troca de dados. Fonte: (IWANITZ; LANGE, 2005)

Critério	Drivers	DDE	OPC
Padronização	-	-	Sim
Interoperabilidade	-	Limitado	Sim
Desempenho	Alto (se otimizado)	Baixo	Alto
Acesso Remoto	-	Sim	Sim (DCOM)
Complexidade	Média	Alta	Alta
Participação no Mercado	Reduzido	Desaparecendo	Predominante

diferentes linguagens de programação, podem não interagir caso uma estrutura padrão não seja utilizada. COM/DCOM é um método de compartilhamento binário de código através de diferentes aplicações e linguagens.

Microsoft Windows permite o compartilhamento de código no nível binário usando DLL (*Dynamic-Link Library* - Biblioteca de Ligação Dinâmica), porém desde que as DLL sejam escritas para uma interface na linguagem C. Desta forma, elas só podem ser utilizadas por aplicações em linguagem C ou por linguagens que entendam suas convenções de troca de mensagens. Este mecanismo coloca a responsabilidade do compartilhamento sobre a construção das linguagens de programação, ao invés de colocá-la na própria DLL.

Microsoft Foundation Class (MFC) é outra alternativa para o mecanismo de compartilhamento binário, uma extensão das DLL. MFC dá aos desenvolvedores de programas um conjunto de componentes reutilizáveis escritos em C++ que encapsulam as funcionalidades necessárias para o desenvolvimento de aplicações para sistemas operacionais Windows. O compartilhamento binário com MFC por sua vez acabou se mostrando restritivo, uma vez que podem ser utilizadas apenas entre aplicações em MFC.

Através da definição de interfaces, a tecnologia COM/DCOM permite que objetos sejam instanciados de forma distribuída, e que seus serviços e métodos (funções) sejam acessíveis por diferentes programas (LIU et al., 2005). Para isso é necessário a utilização de uma linguagem especial, a IDL (*Interface Definition Language* - Linguagem de Definição de Interfaces) (MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION, 1997).

Isto significa que cada cliente pode chamar os métodos de qualquer objeto DCOM em um determinado servidor, independentemente do ambiente de programação que os mesmos foram criados. Através de um identificador único (GUID - *Global Unique Identifier*),

as interfaces são protegidas contra modificações após a sua publicação e a compatibilidade dos objetos DCOM é então garantida (MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION, 1997).

Em DCOM, os componentes *proxy* e *stub* são implementados em uma DLL, utilizando uma IDL que contém a descrição de objetos DCOM e suas interfaces. As interfaces cotem a declaração dos métodos e seus parâmetros, que ao serem compiladas para uma linguagem de programação específica geram os arquivos correspondentes às descrições para esta linguagem.

Estes arquivos quando compilados são ligados, formando uma DLL. A vantagem disto é que a especificação pode ser implementada em qualquer linguagem. Todas as especificações OPC possuem um IDL que é disponibilizada pela OPC Foundation.

A forma de implementação dos servidores determina como os objetos são carregados e gerenciados pelo servidor. Os objetos DCOM são acessíveis através de uma identificação CLSID (*Class Identifier*) mantida pelo registro do sistema operacional (MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION, 1995). Através do CLSID os clientes podem lançar os componentes, solicitar as interfaces dos objetos e chamar os métodos desta interface.

O ciclo de vida de um objeto DCOM é controlado pelo próprio componente (MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION, 1997), de forma que ele gerencia a quantidade de clientes que o referenciam, tornando esta referencia nula quando nenhum cliente está utilizando o objeto, fazendo a liberação dos recursos do sistema (IWANITZ; LANGE, 2005).

Uma desvantagem da tecnologia COM/DCOM é que sua utilização e desenvolvimento dependem de que seja utilizada uma plataforma que as suporte, não garantindo sua portabilidade, sendo naturalmente dependente do sistema operacional Windows (LIU et al., 2005).

2.2 A Arquitetura das Aplicações OPC

O padrão OPC é baseado numa arquitetura cliente-servidor (OPC TASK FORCE, 1998). O paradigma cliente-servidor é usado praticamente em todos os processos distribuídos em que a aplicação servidora aguarda conexões, executa serviços e retorna resultados. Já a aplicação cliente é quem estabelece a conexão com o servidor, envia mensagens para o

mesmo e aguarda pelas respostas (COULOURIS; DOLLIMORE, 2005).

A arquitetura OPC, no entanto, diferencia-se da arquitetura cliente-servidor comum, uma vez que para um cliente existem vários servidores, enquanto é mais comum encontrar um servidor para vários clientes.

Assim, o padrão OPC implementa dois grandes módulos: servidor OPC e cliente OPC (IWANITZ; LANGE, 2005) (OPC TASK FORCE, 1998). O servidor OPC especifica a interface padrão de acesso direto aos equipamentos ou aplicações, disponibilizando os dados dos processos, o cliente OPC especifica a interface padrão para as aplicações terem acesso aos dados coletados (Figura 2).

Um cliente OPC pode se conectar a um ou mais servidores OPC. As interfaces da especificação OPC permitem troca de dados entre componentes de *software* de diferentes fabricantes com grande eficiência.

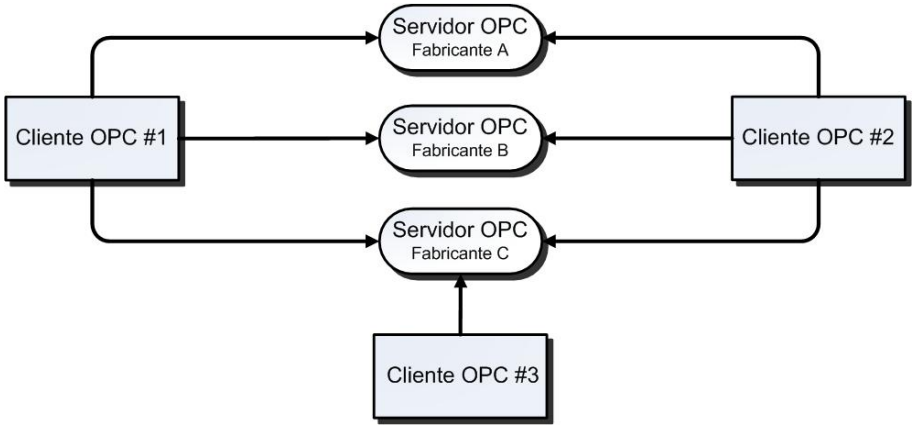

Figura 2: Estrutura Cliente-Servidor OPC. Fonte: (OPC TASK FORCE, 1998)

Existem diversas opções no mercado de clientes e servidores OPC. Em geral cada desenvolvedor de dispositivos fornece seu servidor OPC, que podem ser submetidos à avaliação da OPC Foundation, para verificação se condiz com sua especificação, existindo o mesmo processo para o caso de um cliente OPC.

Numa rede industrial utilizando o padrão OPC, um cliente OPC que pode ser um sistema SCADA comunica-se com o servidor OPC do dispositivo, não tendo que tratar os protocolos de comunicação e formatos dos dados, nem enfrentar conflitos de software e hardware devido a utilização de drivers (Figura 3).

Servidores OPC podem ser acessados por clientes escritos em um grande número de linguagens de programação, incluindo Java, C/C++, Visual Basic, Delphi e linguagens de script (IWANITZ; LANGE, 2005). Os servidores OPC geralmente são escritos em linguagem

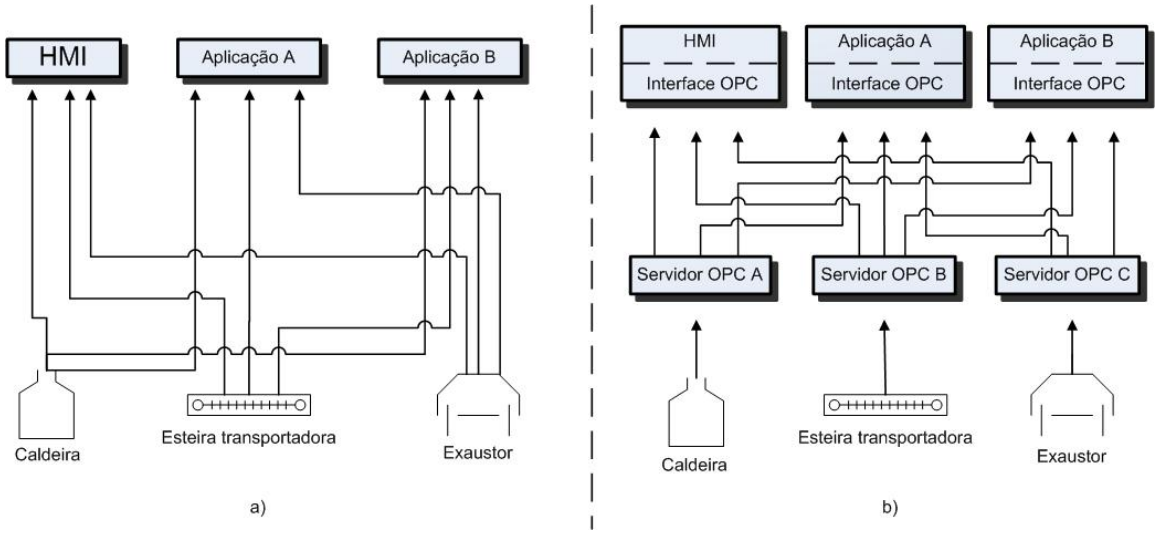

Figura 3: Arquitetura OPC numa rede industrial. a) Solução com drivers. b) Solução OPC

fundamentadas em orientação a objetos como C++ (IWANITZ; LANGE, 2005).

O uso de C++ deve-se principalmente por algumas facilidades encontradas em desenvolver para a plataforma Windows, que é o ambiente mais utilizado em sistemas industriais (ZHENG; NAKAGAWA, 2002) e uma das bases da tecnologia OPC.

Todas as especificações foram feitas de modo a facilitar o desenvolvimento dos servidores OPC e podem também ser escritos em outras linguagens além de C++. O acesso aos servidores OPC é realizado através dos componentes OLE/COM e OLE/DCOM, fornecidos pelo próprio sistema operacional Windows.

O padrão OPC é uma especificação técnica que define um conjunto de interfaces padrão para diferentes tipos de aplicações em tecnologias de automação. OPC Foundation é responsável por dar suporte e liberar novas especificações. Atualmente existem as seguintes especificações publicadas:

- *OPC Overview* - Descrição geral dos campos de aplicação da especificação OPC;
- *OPC Common Definitions and Interfaces* - Definição das funcionalidades básicas para as demais especificações;
- *OPC Data Access Specification* (OPC-DA) - Definição da interface para leitura e escrita de dados em tempo real;
- *OPC Alarms and Events Specification* (OPC-AE) - Definição da interface de monitoramento de eventos;

- *OPC Historical Data Access Specification* (OPC-HDA) - Definição da interface para acesso ao histórico dos dados;
- *OPC Batch Specification* - Definição da interface para acesso aos dados requisitados para processos por batelada (*batch*). Esta especificação é baseada na *OPC-DA Specification* e a estende;
- *OPC Security Specification* - Definição da interface para utilização de políticas de segurança;
- *OPC XML Data Access Specification* (OPC XML-DA) - Definição da integração entre OPC e XML (*Extensible Markup Language*) para construção de aplicações Web;
- *OPC Data eXange Specification* (OPC-DX) - Comunicação entre servidores;
- *OPC Commands Specification* - Definição de uma interface para passagem de comandos e supervisão de sua execução;
- *OPC Complex Data Specification* - Definição da possibilidade de descrever uma estrutura de dados complexos e do acesso a esses dados;
- *OPC Unified Architeture* - Definição de uma plataforma com capacidade de atividade conjunta que padroniza o acesso de dados nos servidores DA, AE e HDA por meio de serviços *Web*.

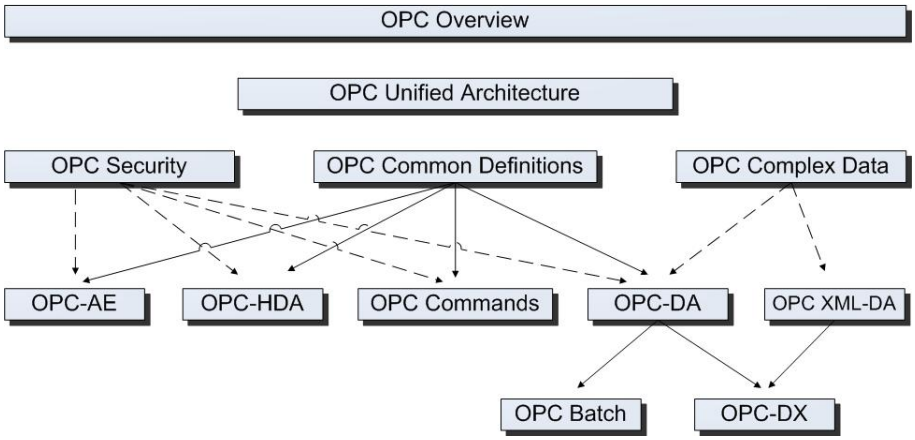

Figura 4: Especificações OPC disponíveis. Fonte: (OPC TASK FORCE, 1998)

A Figura 4 mostra as especificações disponíveis e as relações entre elas. Estas especificações definem diferentes campos de aplicações (servidores) e são dessa forma independentes um dos outros, entretanto é possível combiná-los em uma aplicação (IWANITZ;

LANGE, 2005). As setas com linha cheia representam que a integração de definições entre as especificações é obrigatória. As setas com linha tracejada representam que não há necessidade de integrar as definições, embora seja possível.

Clientes e servidores OPC compartilham alguns componentes em comum. Estes são:

- Arquivos *Proxy/Stub*: permite organizar e desorganizar parâmetros de métodos e ponteiros de interfaces de forma padronizada para que estes possam ser lidos por diferentes aplicações.
- OPCServerBrowser: oferece uma interface que habilita clientes OPC a informarem-se de servidores OPC em computadores remotos.
- Invólucro de Automação (*Automation Wrapper*): permite que classes com interfaces incompatíveis possam interagir, assim um objeto cliente utiliza serviços de outros objetos com interfaces diferentes por meio de uma interface única.

Para garantir que estes componentes estejam disponíveis no computador quando existirem servidores e clientes OPC disponíveis, é necessário que estes componentes sejam criados quando referenciados forem referenciados. Para essa garantia, utiliza-se o conceito de arquivos compartilhados (bibliotecas compartilhadas - *shared libraries*) suportados pelo sistema operacional Windows.

Um contador no registro do sistema operacional é alocado para cada um dos componentes, que após criado é acrescido de um para cada vez que é utilizado, e decrementado para cada vez que é liberado. Se o contador atingir o valor zero significa que não existe um serviço OPC utilizando este componente, sendo então excluído. Por definição da especificação, o fabricante do servidor OPC é responsável pela a instalação dos componentes, sua criação e gerenciamento.

Neste processo de compartilhamento usando o registro do sistema operacional, o cliente precisa de um CLSID do servidor (GUID) para usar o ambiente de execução COM/DCOM para acessar os objetos DCOM (MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION, 1997). Cada servidor OPC deve fazer as três seguintes entradas no registro:

- O identificador do programa (ProgID - *ProgramIdentifier*) contendo uma cadeia de caracteres que descreve o componente. Sua estrutura é predefinida como: Fabricante.Server_name, não podendo conter espaços em branco entre os caracteres.

- Um valor de 128-bit que identifica de forma única um servidor (*ClassID* - CLSID).
- Uma descrição do servidor, contendo informações como permissões de acesso e o nível de autenticação para o servidor (*ApplicationID* - AppID).

Assim, a informação necessária para um cliente acessar um servidor está disponível no registro, sendo uma tarefa para o cliente fazer essa busca no registro, verificando se um recurso está disponível.

A busca poderia ser feita de um computador remoto, entretanto, problemas podem aparecer devido a restrições de acesso. Estes problemas podem ser resolvidos para OPC utilizando o OPCServerBrowser, que é um componente DCOM com CLSID definido na especificação.

Uma vez que o CLSID é definido para o servidor OPC pela especificação, os clientes não precisam percorrer todo o registro. Utilizando o componente OPCServerBrowser, uma interface comum tanto ao cliente quanto ao servidor OPC, eles podem trocar informações, verificando a existência do CLSID, e permite que o cliente obtenha informações do servidor, como o AppID e o ProgID (OPC TASK FORCE, 1998). Essa estrutura de troca de informações usando o componente OPCServerBrowser permite que o cliente acesse o servidor de um computador local ou remoto.

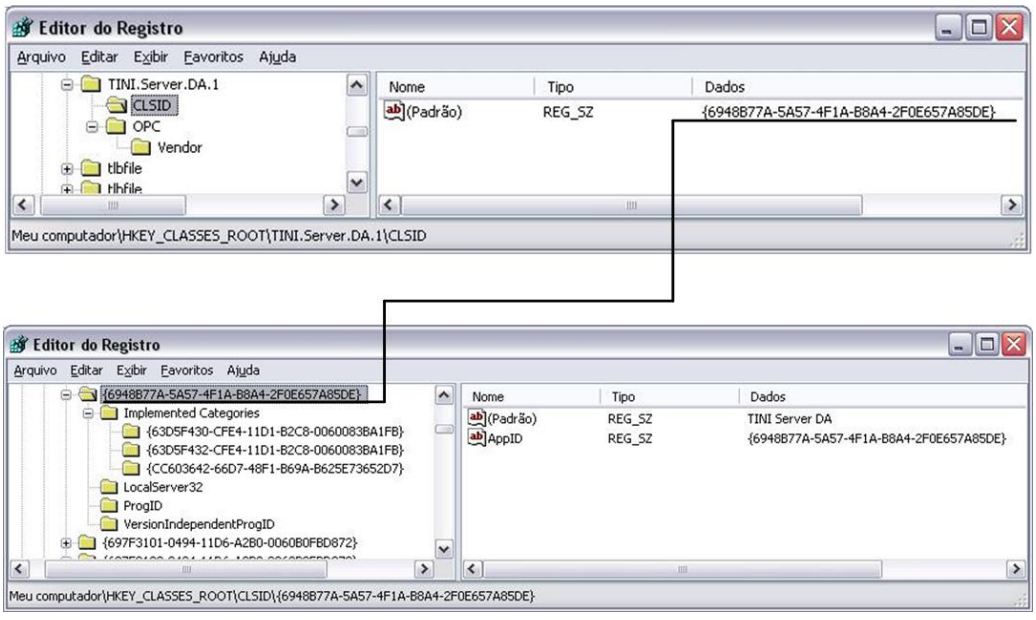

Figura 5: Entradas no registro OPC e suas dependências.

A Figura 5 mostra as diferentes entradas no registro e seus relacionamentos para um servidor OPC. O CLSID mostrado equivale ao servidor para o dispositivo TINI, este

valor esta disponível em outro local do registro, contendo informações sobre o ProgID e o AppID.

Como no caso a implementação é de um servidor *Data Access*, é especificado o valor para a categoria do identificador (CategoryID - CATID) definido na especificação OPC, 6948B77A-5A57-4F1A-B8A4-2F0E657A85DE para a versão 2.0. Os demais CATID mostrados correspondem à versão 1.0 OPC-DA e o servidor de eventos OPC respectivamente.

O componente Invólucro de Automação é uma DLL que permite que classes com interfaces incompatíveis possam interagir através da combinação de interfaces de automação definidas nas especificações OPC. Este componente permite a troca de mensagens dos métodos da interface de automação para a interface padrão.

A Figura 6 mostra a interação entre estas interfaces. Quando a aplicação é de uma linguagem diferente do servidor, o *Automation Wrapper* faz a ligação tornando a comunicação possível.

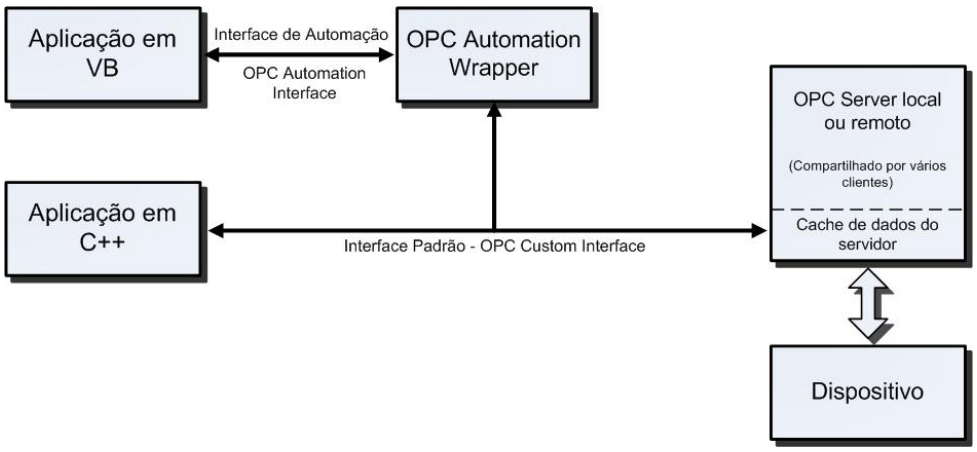

Figura 6: Acesso a um servidor OPC utilizando interfaces padrão e de automação. Fonte: (IWANITZ; LANGE, 2005)

Interfaces padrões são utilizadas em linguagens de programação que suportam os conceitos de ponteiros de função, como C/C++. Existem linguagens de programação, entretanto, que não suportam ponteiros de função, como Visual Basic, e para permitir que clientes OPC escritos nestas linguagens possam acessar os objetos DCOM (OPC), interfaces de automação foram criadas. Interface de automação é um padrão de interface em que os métodos não são acessados por ponteiros de função, mas pelos seus nomes (IWANITZ; LANGE, 2005).

Um objeto de automação implementa propriedades, que representam configurações

destes objetos, que podem acessar o servidor OPC por uma interface padrão. As funcionalidades deste objetos podem ser acessadas pelos clientes, por meio de métodos. Os clientes são informados dos dados do servidor OPC por esse objeto por meio de mensagens (eventos).

Toda a informação sobre os objetos de automação de um componente do servidor são armazenadas na DLL do Invólucro de Automação, permitindo o acesso a métodos, eventos e propriedades, e a interação entre aplicativos de diferentes linguagens de programação.

2.3 Especificação OPC Data Access

A especificação Data Access é a mais antiga das especificações de servidores OPC. Ela define uma interface entre programas cliente e servidor para acesso aos dados do processo. Servidores Data Access permitem o acesso a diferentes fontes de dados de forma transparente aos clientes Data Access, assim como, é possível que os clientes se conectem a diversos servidores ao mesmo tempo. Num sistema SCADA é comum o monitoramento de diversos dispositivos, sendo que um único cliente acessa dados de diferentes servidores.

Um cliente *Data Access* pode ser muito simples, como uma planilha Excel, ou complexo, sendo diversos componentes de programas maiores, como sistemas SCADA e IHM. Servidores Data Access podem ser programas simples, como uma que permita acessar os registradores de um CLP via comunicação serial.

Programas mais complexos também são possíveis, permitindo acessar um grande número de variáveis em uma variedade de dispositivos por mecanismos de comunicação diversos. Servidores Data Access também podem ser componentes de programas maiores, disponibilizando dados a estes.

Um servidor OPC é composto por vários objetos: o servidor, os grupos e os ítens (LING; CHEN; YU, 2004). O objeto servidor mantém informações sobre o servidor e funciona como um container para os objetos de grupo. Os objetos de grupo mantém informações sobre eles mesmos e fornecem o mecanismo para conter e organizar logicamente os ítens OPC. (Figura 7). Os grupos OPC fornecem uma forma dos clientes organizarem os dados.

Um cliente *Data Access* pode criar vários objetos OPC de um servidor para visualizar os processos. O objeto OPCServer é o objeto no topo da hierarquia de objetos OPC. Objetos de grupos (OPCGroup) formam os próximos níveis e os últimos níveis, ou folhas,

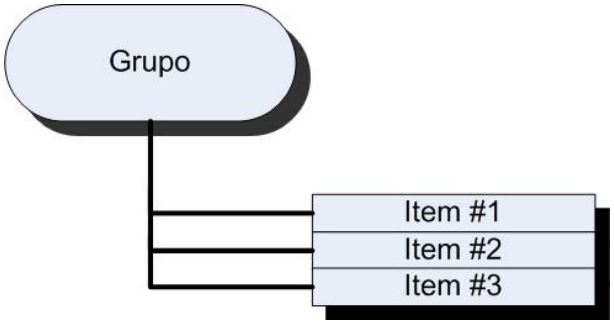

Figura 7: Relacionamento Grupo/Item. Fonte: (OPC FOUNDATION, 2003)

são formados pelos ítens (OPCItem).

Os objetos de grupo são utilizados para agrupar os ítens, de forma a organizá-los quanto o aspecto lógico, agrupando-os de acordo à sua composição física, (todos os ítens relacionados, por exemplo, às válvulas estão em um grupo), ou quanto às funcionalidades (os ítens destinados a monitorar um processo, como a temperatura de uma caldeira, estão no mesmo grupo). Todos os objetos de grupo são monitorados pelo objeto OPCServer.

Somente dois objetos da hierarquia de objetos são objetos DCOM reais - OPCServer e OPCGroup. Para estes dois objetos, interfaces, métodos, e parâmetros foram definidos (Figura 8). O objeto OPCItem não possui interfaces adaptáveis, pois, numa aplicação, vários valores são lidos e escritos ao mesmo tempo, sendo mais eficiente acessar os objetos ítem com chamadas simples (IWANITZ; LANGE, 2005). Estas chamadas são feitas através de métodos das interfaces dos objetos de grupos (OPCGroup), não excluindo o acesso a leitura e escrita dos objetos OPCItem.

A implementação deste padrão proverá um servidor OPC que possibilita a conexão de sistemas SCADA com um sistema embarcado baseado na plataforma desenvolvimento TINI.

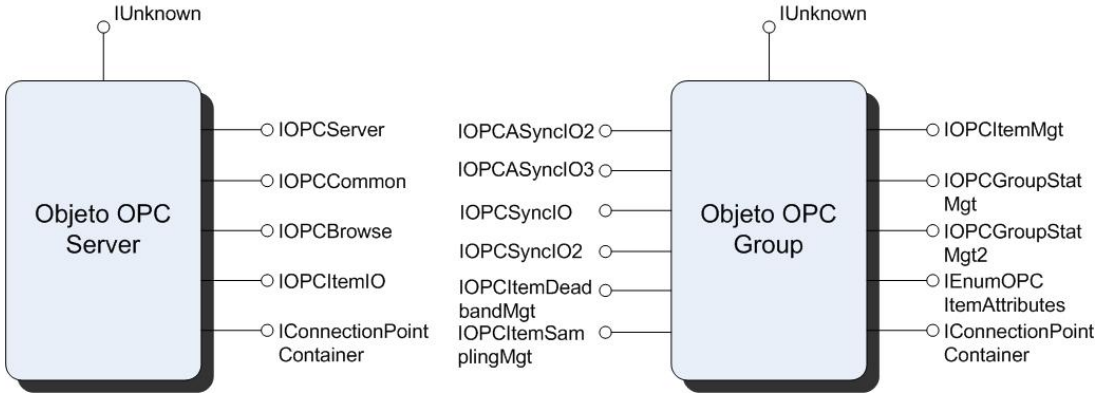

Figura 8: Componentes OPCServer e OPCGroup. Fonte: (OPC FOUNDATION, 2003)

OPC Data Access fornece a base da funcionalidade de acesso a dados de diversos dispositivos em uma rede, através de um conjunto de interfaces padrão. A intenção principal da especificação *OPC Data Access* é prover a aquisição de dados dos dispositivos através destas interfaces e que estes possam ser dispostos a um cliente OPC (OPC FOUNDATION, 2003).

Nesta especificação, são definidos dois diferentes conceitos: a hierarquia dos objetos OPC e o espaço de nomes (namespace). Um espaço de nomes é um recipiente abstrato que fornece um contexto aos ítens, representando todas as fontes de dados e depósito de dados disponíveis no servidor (IWANITZ; LANGE, 2005).

O espaço de nomes pode ser uma estrutura de árvore hierárquica, onde as folhas representam os dados, valores mensuráveis de um determinado dispositivo. Estas folhas são os ítens OPC ou tags (OPC FOUNDATION, 2003). (Figura 9)

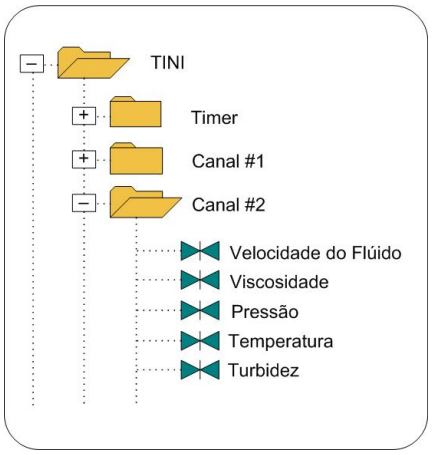

Figura 9: Exemplo de namespace OPC.

Cabe ao desenvolvedor do servidor determinar os níveis de hierarquia de objetos OPC para seu dispositivo, o qual após definido é fixo. A Figura 10 mostra a hierarquia de objetos e seus relacionamentos em um *namespace*.

Para um servidor, existe apenas um *namespace* que é o mesmo para todos os clientes. O cliente mostrado pode acessar no máximo os dois grupos do servidor, nos quais são definidos dois objetos item para cada, acessando as folhas da árvore hierárquica.

Em uma aplicação, não somente os dados dos processos do dispositivo são importantes, mas também informações sobre essas variáveis e sobre os próprios dispositivos. Disponibilizar estas informações em ítens expande de forma desnecessária o *namespace*.

Assim, estes atributos, chamados de propriedades, são alocados nos nós e folhas a que correspondem. O nome de um fabricante de dispositivo pode ser mapeado como uma pro-

priedade, em geral atribuído ao nó raiz da hierarquia de nomes OPC (OPC FOUNDATION, 2003).

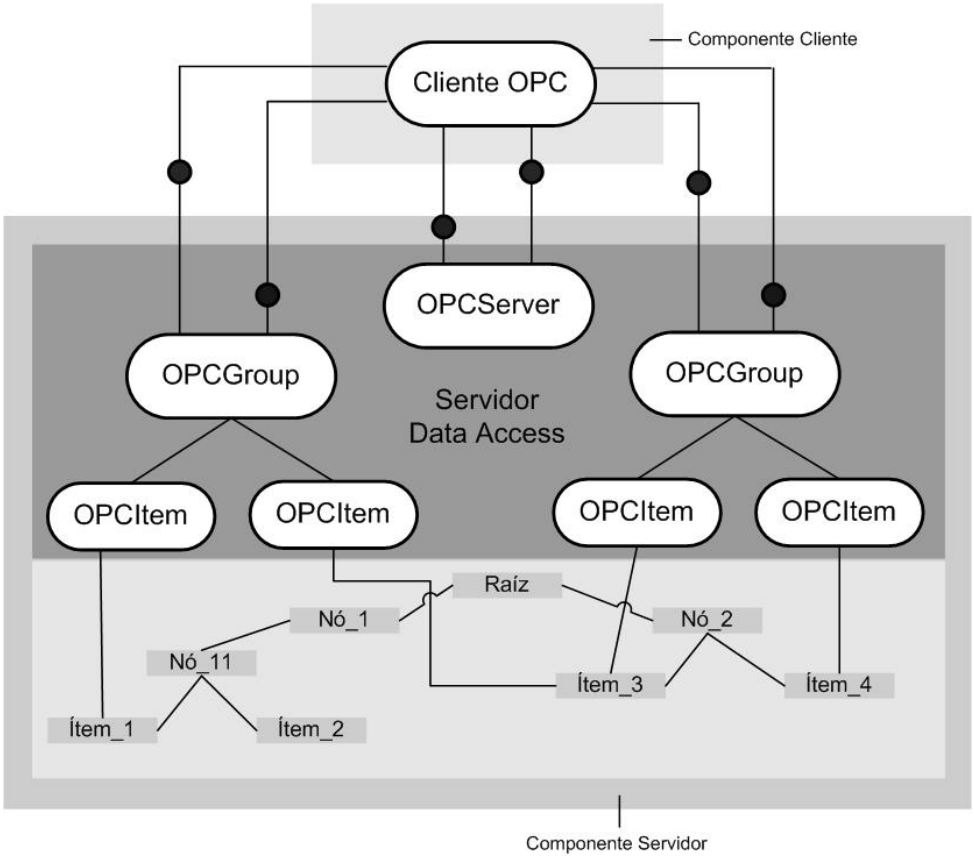

Figura 10: Namespace e hierarquia de objetos em um servidor OPC-DA. Fonte: (IWANITZ; LANGE, 2005)

Além das propriedades, informações sobre os caminho de acesso (AccessPath) a um item pode estar disponível nele, porém, é opcional e pode ser omitido. Pode ser usado para descrever o caminho da comunicação entre o servidor *Data Access* e o dispositivo de forma detalhada, como por exemplo, que o dispositivo esta conectado na COM1 a 9600KBits/s.

A especificação define ainda um conjunto de interfaces que permitem a utilização de comunicação síncrona e assíncrona (OPC FOUNDATION, 2003) com os dispositivos de *hardware*. A troca de dados entre o servidor e o cliente OPC pode ser feita através do acesso à cache interna do servidor sobre os dados do processo (cache) ou diretamente da fonte de dados, o dispositivo (Tabela 2).

Três possibilidades básicas podem ser distinguidas para como o acesso aos dados é realizado pelo servidor *Data Access*: os valores estão disponíveis no próprio computador (outra aplicação), ou devem ser requisitados de dispositivos externos, ou são enviados por

dispositivos externos sem que sejam requisitados (IWANITZ; LANGE, 2005).

Na leitura síncrona, o cliente chama um método e espera pelo retorno do valor. Chamadas síncronas são utilizadas somente se o acesso aos dados é feito de forma rápida, como valores disponíveis no computador, para que a performance do cliente não seja afetada pela espera da resposta à sua chamada.

Com a leitura assíncrona, o cliente chama o método do servidor e espera pelo retorno, depois de certo intervalo de tempo, que depende do tipo de dado, o cliente obtém o valor desejado. A leitura assíncrona deve ser utilizada se a fonte de dados está distante, e os valores devem ser requisitados.

Tabela 2: Tipos de troca de dados entre servidor e cliente OPC-DA. Fonte: (IWANITZ; LANGE, 2005)

Tipo de troca de dados	Cache	Dispositivo
Leitura síncrona	x	x
Leitura Assíncrona		x
Atualização (refresh)	x	x
Subscrição		

Com a atualização (refresh), o cliente obtém todos os objetos OPCItem ativos de um OPCGroup. Para os três tipos de trocas de dados citados, o cliente sempre toma a iniciativa.

Estas três formas nem sempre são eficientes, pois o cliente sempre estará solicitando dados sem se preocupar com alguma possível mudança do valor do dado. Por esta razão, existe uma quarta forma de troca de dados, no qual o servidor obtém os dados dos dispositivos a uma determinada freqüência e repassa este valor ao cliente apenas se houver alguma mudança (subscrição) (OPC FOUNDATION, 2003) (IWANITZ; LANGE, 2005).

Os dados trocados entre o cliente e o servidor OPC consistem de três componentes: o dado atual, um carimbo de tempo e o estado da informação (Figura 11). O dado atual representa o dado lido do dispositivo, associado a ele o tipo de dado. O carimbo de tempo é o uma estrutura padrão de data/hora para a representação das coordenadas de tempo universais. O estado da informação é dividido em três partes: qualidade do valor, estado do dispositivo e diagnostico de falhas.

A qualidade do valor pode ser "*Good*", "*Bad*", ou "*Uncertain*". O servidor pode dispor a qualidade do valor como "Bad", por exemplo, se o servidor não está conectado

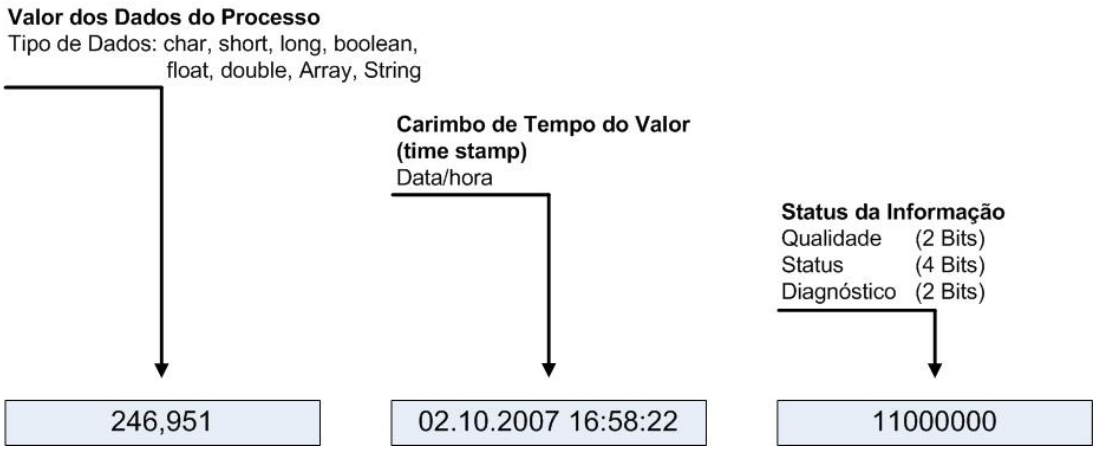

Figura 11: Formato dos dados para a especificação OPC-DA. Fonte: (IWANITZ; LANGE, 2005)

ao dispositivo. O status do dispositivo contém valores como "*Not Connected*"para um dispositivo não conectado ao servidor (IWANITZ; LANGE, 2005).

As interfaces propostas pela especificação permitem que um cliente se conecte a um servidor OPC. O servidor OPC fornece funcionalidades ao cliente OPC para criar e manipular grupos de objetos.

Estes grupos permitem aos clientes organizarem os dados e o modo como acessá-los. Um grupo pode ser ativado ou desativado como uma unidade. Um grupo também fornece mecanismos de um cliente subscrever uma lista de ítens, sendo notificado quando estes mudarem.

Uma das limitações em utilizar um servidor OPC-DA é no que diz respeito a capacidade de transferir dados do servidor para o cliente pelo meio mais fácil e eficiente, o que é um agravante devido ao grande fluxo de dados que por vezes se encontram numa rede industrial. Como alternativa, pode-se utilizar o mecanismo de caching de dados localizados nos clientes (VERYHA, 2005).

3 Desenvolvimento do Sistema

Este trabalho relata o desenvolvimento de um servidor OPC que possibilita a conexão de sistemas SCADA a um sistema embarcado baseado na plataforma TINI. O estabelecimento de um servidor OPC para a plataforma TINI possibilita integrar um dispositivo Ethernet em um ambiente de automação industrial, sem que seja apenas para soluções específicas e dedicadas.

Para projetar esta solução, segmentou-se o sistema em três partes para o desenvolvimento e o estudo destas: o sistema supervisório, o servidor OPC (COM/DCOM) e a implementação do padrão de comunicação entre o dispositivo e o servidor OPC, tornando os dados do dispositivo acessíveis.

Através desta metodologia, pode-se identificar os componentes do sistema (Figura 12), englobando os dispositivos de hardware a serem utilizados e a forma de conexão entre o dispositivo e o servidor. Os clientes da solução são ferramentas SCADA, que também são cliente OPC, oferecendo uma interface homem-máquina e a exibição de valores dos ítens do servidor TINI OPC-DA.

Como o meio de comunicação a ser utilizado para a transmissão de dados entre o dispositivo e o servidor OPC é a Ethernet, e como o dispositivo implementa os protocolos da camada de rede (IP) e de transporte (TCP), optou-se o estabelecimento da conexão utilizando *sockets*.

Um *socket* é definido como uma abstração computacional que mapeia diretamente uma porta de transporte e um endereço de rede, estabelecendo um elo bidirecional de comunicação entre dois programas. *Sockets* são diferenciados em cliente e servidor.

Realizar uma conexão via *socket* baseia-se no fato de que um dispositivo aguarda uma conexão em uma determinada porta (servidor) e que outro tenta se conectar ao primeiro na determinada porta (cliente). Nesta solução, o cliente *socket* é o servidor OPC instalado em um computador e o servidor *socket* é a plataforma TINI.

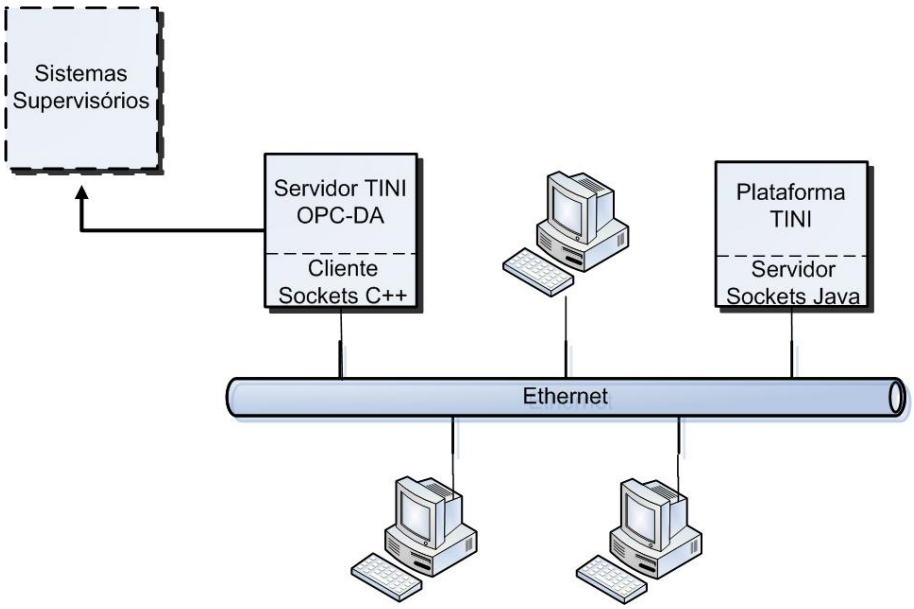

Figura 12: Componentes do Sistema.

De acordo aos requisitos de funcionalidade do sistema, escolheu-se a linguagem C++ para a construção do servidor OPC, uma vez que na literatura encontrou-se mais suporte e exemplos para a implementação de servidores COM/DCOM nesta linguagem.

Como a solução desenvolvida é voltada para o sistema operacional Microsoft Windows, adotou-se a ferramenta *Microsoft Visual C++ v6.0* (MSVC++) para o desenvolvimento do *software*. MSVC++ da suporte à Interface de Programação de Aplicativos da *Microsoft Developer Network* (MSDN) possibilitando acessar algumas funcionalidades do sistema operacional, além de facilidades em utilizar a tecnologia COM.

Por sua vez, como a plataforma TINI dá suporte à linguagem Java, esta linguagem foi utilizada para desenvolver o servidor socket e para disponibilizar alguns de seus recursos a serem transmitidos para o cliente socket.

3.1 Sistema Supervisório

Para o sistema supervisório, utilizou-se ferramentas de simulação da Matrikon Inc. e da Wonderware FactorySuite para compreender a estrutura de um sistema SCADA e o padrão de conectividade definido pela OPC Foundation, assim como entender os conceitos de troca de dados usando esta tecnologia.

As ferramentas *MatrikonOPC Server Simulation v1.2.4.1* e *MatrikonOPC Explorer v3.3.2.0* funcionam, respectivamente, como servidor e cliente OPC. Através destas ferra-

mentas pode-se compreender a estrutura de servidor, grupo e ítem OPC, o formato dos dados para a especificação OPC-DA, a hierarquia de objetos e o espaço de nomes OPC.

O MatrikonOPC Explorer serve como cliente pode ser utilizado como cliente para qualquer servidor OPC instalado em um computador, acessando o registro e os identificando de acordo ao CLSID das especificações. Esta ferramenta foi utilizada como cliente para testes e verificação de dados do servidor TINI OPC-DA.

A ferramenta *Wonderware OPCLink I/O Server v8.0* provê um canal de comunicação entre um servidor OPC e os ítens contidos nele para as demais aplicações da Wonderware. Uma destas aplicações é a *Wonderware Intouch v9.5* que permite que uma interface homem-máquina seja construída para gerenciamento de um processo de automação.

Utilizando o MatrikonOPC Server Simulation com as ferramentas da Wonderware, ítens OPC de um servidor podem ser exibidos em uma interface homem-máquina. Ao substituir o servidor OPC simulado pelo TINI OPC-DA, testou-se soluçõo desenvolvida, contemplando a conexão do dispositivo com um sistema SCADA.

3.2 Servidor OPC-DA

3.2.1 Servidor COM/DCOM

Um objeto ou componente COM é uma estrutura que disponibiliza suas funções através do mecanismo de interfaces. Em aplicações em C++, interfaces são definidas como classes abstratas, ou seja, somente descreve a assinatura das funções para alguma outra classe implementá-las. A declaração da interface é feita utilizando um arquivo IDL, formato este independente da linguagem de programação.

Ilustrando os conceitos que abordam essa tecnologia, esta seção descreve os passos básicos para a implementação de um servidor COM, sem a utilização de MFC. Estes passos foram utilizados durante a metodologia de aprendizagem, para posterior construção de um servidor OPC-DA. O processo de construir um servidor COM pode ser dividido em seis passos básicos.

1 Criação o arquivo IDL correspondente ao servidor

Um arquivo IDL contém uma descrição genérica das interfaces que um dado componente de software pode conter, associado a um GUID (Listagem 3.1). Para a gerar um GUID foi utilizada a ferramenta da Microsoft GUIDGen v1.0. Esse valor

refere-se ao CLSID do servidor.

Listagem 3.1: IAdicao.idl

```
1 import "unknwn.idl";

3 [object,
uuid(307BA609-C95A-469f-99EE-4BD017B4AD57),
5 helpstring("interface do servidor COM")
]
7
interface IAdicao : IUnknown{
9     HRESULT       PrimeiroNumero(long nX1);
      HRESULT       SegundoNumero(long nX2);
11    HRESULT       OpAdicao([out,retval] long *pBuffer);
};
```

HRESULT é um tipo de dado para indicar o estado da operação de componentes Microsoft que possui 32 bits. Um valor do tipo HRESULT possui três campos: código de sucesso da operação, código identificador da operação, e um código de erro. O código de sucesso da operação indica se o valor de retorno representa informação, advertência ou erro. O código identificador da operação identifica a área do sistema responsável pelo erro. O código de erro é um valor único que identifica uma exceção. Quando uma exceção ocorre, o valor de HRESULT é passado ao cliente COM (MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION, 2000).

2 Geração das bibliotecas de acordo à linguagem de programação

Para gerar as bibliotecas para a interface do servidor COM, utilizou-se o programa MIDL v1.0 (Microsoft IDL) que compila o arquivo IDL para a linguagem C++. Após a compilação, foram gerados os arquivos:

- IAdicao.h - Contém a declaração das interfaces, para a linguagem C++.
- dlldata.c - Contém o código-fonte do proxy do servidor DCOM, utilizada quando o objeto é invocado em um computador diferente.
- IAdicao.tlb - Arquivo binário que descreve a interface IAdicao, distribuída aos clientes COM.
- IAdicao_p.c - Contém o código-fonte para construção do proxy do servidor.

- IAdicao_i.c - Contém o identificador de classe.

3 Construção das classes derivadas das interfaces do servidor

Após gerar as bibliotecas, é necessário implementar as interfaces descritas, para criar um objeto COM. Para isso criou-se dois novos arquivos, AdicaoObj.h e a classe em C++ correspondente CAdicaoObj.

AdicaoObj.h por sua vez, deve ser derivado de IAdicao.h, também derivando de IUnknown, que é a classe abstrata base. Assim, Adicao.h deve declarar todos os métodos das classes abstratas bases de quem herda (Listagem 3.2).

Listagem 3.2: AdicaoObj.h

```
#include "IAdicao.h"
extern long g_nComObjsInUse;
class CAdicaoObj : public IAdicao
{
    public:
    //IUnknown interface
    HRESULT __stdcall QueryInterface(
                                REFIID riid ,
                                void **ppObj);
    ULONG   __stdcall AddRef();
    ULONG   __stdcall Release();

    //IAdicao interface
    HRESULT __stdcall PrimeiroNumero(long nX1);
    HRESULT __stdcall SegundoNumero(long nX2);
    HRESULT __stdcall OpAdicao([out,retval] long *pBuffer);

    private:
    long m_nX1 , m_nX2;
    long m_nRefCount;
}
```

A variável m_nRefCount foi utilizada para gerenciar a referencia ao servidor, atendendo à especificação COM. O método QueryInterface da interface IUnknown é utilizado para passar um ponteiro a um cliente que deseja utilizar o servidor, permitindo que os clientes acessem as demais interfaces (IAdicao). AddRef e Realease

são utilizados para incrementar e decrementar a referencia ao servidor feita por clientes.

4 Implementação dos métodos das interfaces

Após a construção das classes derivadas da interface, implementou-se os métodos dessa interface. A implementação da interface provê um mecanismo simples para a soma de dois números e os requisitos básicos de funcionamento do servidor COM. A Listagem 3.3 mostra parte do código-fonte da classe CAdicaoObj, para a interface IUnknown.

Listagem 3.3: Implementação da IUnknown em AdicaoObj.cpp

```
1  HRESULT __stdcall CAddObj::QueryInterface(REFIID riid,
                                              void **ppObj)
3  {
       if (riid == IID_IUnknown){
5          *ppObj = static_cast(this);
           AddRef() ;
7          return S_OK;
       }
9      if (riid == IID_IAdd){
           *ppObj = static_cast(this);
11         AddRef();
           return S_OK;
13     }
       *ppObj = NULL ;
15     return E_NOINTERFACE;
   }
17
   ULONG    __stdcall CAddObj::AddRef(){
19     return InterlockedIncrement(&m_nRefCount);
   }
21
   ULONG    __stdcall CAddObj::Release(){
23     long nRefCount=0;
       nRefCount=InterlockedDecrement(&m_nRefCount);
25     if (nRefCount == 0) delete this;
       return nRefCount;
```

```
27 }
```

5 Implementação da interface IClassFactory

De acordo à especificação COM, todo objeto deve ter uma implementação separada da interface IClassFactory. Os clientes COM utilizam essa interface para obter uma instância da implementação da interface do servidor criado, no caso o correspondente à interface IAdicao.

Listagem 3.4: Implementação da IClassFactory em AdicaoObjFactory.cpp

```
1 HRESULT __stdcall CAddFactory::CreateInstance(
                                    IUnknown* pUnknownOuter,
3                                   const IID& iid, void** ppv)
  {
5     // Classes não podem ser agregadas.
      if (pUnknownOuter != NULL){
7         return CLASS_E_NOAGGREGATION;
      }
9     // Componente não pode ser criado na memória.
      CAddObj* pObject = new CAddObj;
11    if (pObject == NULL){
          return E_OUTOFMEMORY;
13    }
      // Obtém a interface.
15    return pObject->QueryInterface(iid, ppv);
  }
17
  HRESULT __stdcall CAddFactory::LockServer(BOOL bLock){
19    return E_NOTIMPL;
  }
```

Assim, além da implementação dos métodos da interface IUnknown, a classe criada implementa os métodos LockServer e CreateInstance. O processo para criação da classe para construção dos métodos destas interfaces é análogo ao da criação da classe AdicaoObj.cpp. O método CreateInstance provê um mecanismo de controle para criação de uma instância do componente COM por um cliente. O método LockServer permite o controle de habilitar ou não a criação do servidor (Listagem

3.4)

A agregação faz-se necessário quando duas classes associadas têm um sentido próprio e separadas continuam existindo como unidade autônoma, podendo até se associar com outras instâncias. Quando isso não ocorre, não e permitido que o cliente acesse o servidor (MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION, 1995).

6 Registro do componente

Após a implementação das interfaces COM e a compilação do projeto utilizando o MSVC++, as DLL necessárias foram criadas, incluindo o proxy, através do qual o cliente pode criar uma instância do servidor. Para que a operação seja realizada, é necessário que o servidor esteja no registro do sistema operacional.

Esse registro pode ser feito através da implementação em C++, ou através de um arquivo simples de extensão "reg"que escreve na chave HKEY_CLASSES_ROOT o nome do servidor (ProgId) e o CLSID, e na chave HKEY_CLASSES_ROOT/CLSID/ 307BA609-C95A-469f-99EE-4BD017B4AD57 o AppID e o ProgID. Observa-se que o valor do CLSID do servidor deve ser o mesmo utilizado em sua criação, caso contrario, os valores serão diferentes do proxy, impossibilitando que o cliente acesse o servidor.

3.2.2 Desenvolvimento do servidor OPC-DA

O desenvolvimento do servidor OPC-DA teve início utilizando os procedimentos para construção de um servidor COM, construindo as interfaces dos componentes OPCServer e OPCGroup conforme visto na Figura 8, utilizando MFC e a documentação da MSDN.

Como arquivos IDL, para cada uma das especificações, a OPC Foundation disponibiliza um conjunto da descrição destes componentes que está na versão 3.00 (OPC Core Components - Núcleo de Componentes OPC). Estes arquivos IDL podem ser utilizados de acordo à necessidade da construção do servidor por parte do desenvolvedor. Além dos arquivos IDL, são fornecidos os arquivos DLL do proxy e stub dos servidores.

Assim, a partir do procedimento de construção de um servidor COM citado anteriormente, utilizou-se os arquivos IDL fornecidos para implementação das interfaces. Este procedimento mostrou-se ineficaz, uma vez que a documentação da especificação apenas explica em linhas gerais quais são as interfaces e suas funcionalidades do OPC-DA, assim como a documentação de COM/DCOM explica apenas o funcionamento básico

destes. Como resultado do processo, apenas foi desenvolvido de forma funcional a estrutura grupo/ítem.

Além dos problemas ao lidar com a especificação, a utilização de MFC e a documentação da MSDN dificultou um pouco o processo, inclusive para a implementação da comunicação com o TINI, pois exemplos disponibilizados pela MSDN não eram funcionais. Apesar de apresentar algumas facilidades ao utilizá-los, a programação usando a Interface de Programação de Aplicativos do sistema operacional Microsoft Windows não é algo usual, com referencia bibliográfica distinta da MSDN difícil de ser encontrada.

Neste desenvolvimento, percebeu-se que algumas das funcionalidades para construção da estrutura de espaço de nomes, disponibilização dos ítens e suas propriedades, a criação e encerramento do servidor eram construídos de forma padrão, sendo igual para qualquer servidor.

Desta forma, buscou-se alguma ferramenta específica, que a partir dos arquivos IDL, gerassem uma estrutura básica para utilizar as funcionalidades OPC, sem a necessidade do procedimento da construção das interfaces. A ferramenta Softing OPC Toolbox C++ v4.1 da empresa alemã Softing foi a única opção encontrada, sendo adquirida.

Esta ferramenta, a partir do Núcleo de Componentes OPC, constrói as interfaces da especificação OPC-DA, entre outras, oferecendo o acesso às funcionalidades citadas através de funções com uma API própria. Um inconveniente ao utilizar esta ferramenta é que as classes geradas são incompatíveis com MFC, não sendo aproveitado o trabalho anterior, inclusive a comunicação por *sockets* desenvolvida para acesso ao dispositivo.

O tipo de comunicação escolhida para o servidor foi assíncrona, de forma que a performance dos clientes não sejam afetadas, esperando por um dado requisitado. O acesso aos dados do servidor, podem ser feitos de duas formas: direta ou da fila de eventos.

A Listagem 3.5 mostra a criação do espaço de nomes para o TINI, utilizando o tempo local do dispositivo (data/hora) sendo disponibilizados diretamente ou da fila de eventos. Além disso, pode ser observado a inserção nos ítens (*tag*) as propriedades referentes aos mesmos.

Listagem 3.5: Implementação do espaço de nomes do servidor OPC-DA

```
BOOL CServerApp::buildObjectSpaces(void){
2       SOCmnPointer<SODaSEntry> entryDA = getSODaSEntry();
        SOCmnPointer<SODaSNameSpaceRoot> nsRoot =
4                               entryDA->getNameSpaceRoot();
```

```
6           // Espaço de nomes do servidor OPC-DA
            SOCmnPointer<SODaSTag> tag;
8           SOCmnPointer<SODaSProperty> prop;

10          tag = new CTag();
            tag->setDeviceIOMode(SODaSItemTag::queue);
12          tag->setName(_T("Tempo_Fila"));
            tag->setAccessRights(OPC_READABLE);
14          tag->setNativeDatatype(VT_BSTR);
            nsRoot->addLeaf(tag);
16
            prop = tag->addProperty(1);
18          prop->setConstantValue(SOCmnVariant(_T("Modo_fila")));

20          tag = new CTag();
            tag->setDeviceIOMode(SODaSItemTag::direct);
22          tag->setName(_T("Tempo_Direto"));
            tag->setAccessRights(OPC_READABLE);
24          tag->setNativeDatatype(VT_BSTR);
            nsRoot->addLeaf(tag);
26
            prop = tag->addProperty(2);
28          prop->setConstantValue(SOCmnVariant(_T("Modo_direto")));

30          return TRUE;
}
```

Os valores para a propriedade (Linhas 17 e 27) são apenas um identificador para a propriedade daquele ítem, podendo até mesmo possuir o mesmo valor de identificador, pois são ítens diferentes.

Para a comunicação com o dispositivo, implementação do cliente socket, foi estabelecido a troca de dados através de caracteres apenas. Esta escolha deve-se à incompatibilidade do tamanho dos tipos de dados entre Java (servidor) e C++. Desta forma, o cliente *socket* solicita ao dispositivo o dado que deseja, entrando em modo de espera por uma

resposta do dispositivo.

O dado estabelecido para testes é o do relógio do TINI, que provê informações de data e hora em tempo real, podendo ser utilizado também como uma carimbo de tempo de um valor qualquer do dispositivo.

Assim, para o caso de teste, estabeleceu-se um protocolo simples de comunicação. O servidor OPC envia o caractere "d"ao dispositivo. O dispositivo por sua vez, entende que o caractere "d"representa que o cliente socket deseja que seja enviado o valor data e hora do dispositivo. Apenas por padrão, a solicitação feita pelo servidor OPC é feita a cada 10 segundos.

3.3 Plataforma TINI

Tiny InterNet Interface (TINI) é uma plataforma desenvolvida pela Dallas Semiconductor que combina um microcontrolador (DS80C400) com interface Ethernet e o ambiente de programação em Java (JDK - *Java Development Kit*), assim provendo processamento, controle e capacidade de comunicação em rede, expondo o acesso ao *hardware* através de um conjunto de interfaces em Java.

A placa utilizada foi o modelo TBM400, composta por um soquete com o microcontrolador DS80C400 e um controlador Ethernet, e por uma placa de encaixe para o soquete que disponibiliza o acesso às portas de entrada e saída e alimentação do *hardware*.

Listagem 3.6: Implementação do servidor socket para o TINI.

```
1 while (clientSocket.isConnected() == true) {
    if(a=='d'){
3           DataOutputStream dado = new
                DataOutputStream(clientSocket.getOutputStream());
5           Clock rtc = new Clock();
            inputMsg = rtc.getDate() + "/" + rtc.getMonth() +
7               "/" + rtc.getYear() + "/" + "␣" + rtc.getHour() +
                ":" + rtc.getMinute() + ":" + rtc.getSecond();
9           dado.writeChars(inputMsg);
    }
11 }
```

O software desenvolvido para a plataforma provê um mecanismo simples de comu-

nicação com o servidor OPC utilizando *sockets*. A Listagem 3.6 mostra a solução desenvolvida, bem como a montagem do carimbo de tempo (Linhas 6 a 8).

Para que a solução desenvolvida atenda à versão do software de controle de *hardware* (*firmware*) do TINI, é necessário que o código-fonte compilado seja convertido para que possa ser interpretado de acordo aos requisitos do hardware. Para isso, utiliza-se o utilitário TINIConverter que converte os arquivos "class"Java para uma imagem binária de extensão "tini"capaz de ser executada no dispositivo.

4 Resultados

Durante o processo de desenvolvimento da solução, dividiu-se o projeto em duas partes: a implementação do servidor OPC-DA e da comunicação entre o servidor e o dispositivo.

Para a implementação do servidor OPC-DA, foi desenvolvido um primeiro protótipo, sem estar conectado ao dispositivo, para testar as funcionalidades e conceitos OPC, e os mecanismos para utilização deste com os supervisórios (clientes). Para este protótipo, simulou-se a utilização de diversos tipos de dados e construindo o espaço de nomes OPC.

Durante a realização do projeto, uma das dificuldades encontradas para a construção do servidor foi a pouca disponibilidade de bibliografia referente à implementação das especificações OPC. Utilizar os simuladores para testes e a ferramenta Softing OPC Toolbox C++ v4.1 foi importante para o desenvolvimento do projeto.

Figura 13: Pataforma TINI.

Estes testes possibilitaram a validação do servidor, permitindo o avanço do projeto, no qual foi desenvolvida a comunicação com o dispositivo e a disponibilização de dados internos (data e hora do relógio interno do TINI). A Figura 13 mostra a plataforma TINI conectada utilizando a interface Ethernet. O dado referente á data e hora do relógio pode ser monitorado também utilizando a interface RS-232.

Após a construção do servidor e do mecanismo de comunicação, as soluções propostas foram integradas, verificando as inconsistências como a troca de dados entre diferentes lin-

guagens de programação, C++ e Java, devido a tratarem de forma diferente os tamanhos dos tipos de dados.

Figura 14: Janela principal da solução de servidor OPC-DA para o TINI.

Como solução, apenas caracteres são trocados entre o servidor e o dispositivo no corresponde aos dados, assim o servidor solicita a um intervalo de 10 segundos que o dispositivo mande informações de seu relógio interno. O dispositivo é acessado pelo servidor através de uma interface, na qual são informados o número IP e porta do dispositivo, sendo possível conectar-se e desconectar-se deste (Figura 14). Por padrão, adotou-se como endereço de rede para acesso ao dispositivo o IP 192.168.0.15 e porta 1050.

A solução proposta foi utilizada com dois clientes OPC distintos. A Figura 15 mostra a conexão do dispositivo Ethernet ao supervisório através do servidor OPC-DA.

Figura 15: Teste com o supervisório da Matrikon.

Na Figura 15, o servidor está conectado ao cliente MatrikonOPC Explorer, através do qual pode ser visto o valor do dado utilizando acesso direto ou à fila de eventos, o carimbo de tempo, a qualidade e o status da informação. O valor é solicitado a cada 10

segundos ao servidor, que por sua vez busca a informação no dispositivo.

Utilizando o Wonderware OPCLink I/O Server, o servidor implementado para a plataforma TINI foi reconhecido, assim como o seus ítens. Os ítens reconhecidos, são incluídos no dicionários de ítens do Wonderware Intouch (IHM), sendo também reconhecidos como ítens OPC.

Quando uma interface é construída, utilizando os ítens do dicionário deste *software*, o ítem OPC é adicionado à IHM, porém não é permitido que esta seja utilizada para simulação em tempo de execução, pois a versão do *software* não é licenciada.

Assim, as ferramentas da Wonderware, permitiram que os ítens OPC fosse adicionados aos seus dicionários de ítens, porém, não permitido adicioná-los para serem utilizados em sua IHM, pois trata-se de ferramentas proprietárias, exigindo uma licença do proprietário dos softwares para utilizar os novos ítens de seus dicionários.

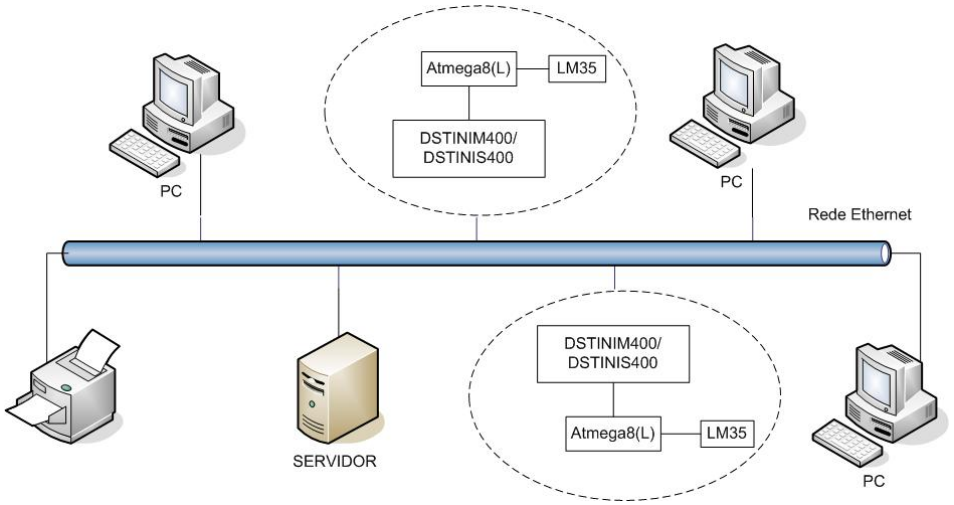

Figura 16: Proposta do protótipo de automação industrial com a plataforma TINI.

Após os testes de validação do servidor com os simuladores, o projeto tinha como objetivo a construção de um protótipo de automação industrial utilizando a plataforma TINI. Para isso, seria conectado um microcontrolador ATMEGA8(L) através da interface SPI (*Serial Peripheral Interface*). O microcontrolador transferiria dados de temperatura adquiridos do sensor LM35 através do conversor analógico-digital (Figura 16).

A construção deste protótipo foi construída em trabalho anterior de iniciação científica intitulado "Uma Interface em Plataforma TINI para Conexão de Redes Ethernet com Sistemas Embarcados"(PIBIC/CNPq 2005). A integração do protótipo com o servidor OPC para o TINI não foi testada devido a falta de componentes em laboratório de graduação (Labhard).

5 Conclusão

Neste trabalho foram apresentados a descrição e o funcionamento de um sistema capaz de permitir a conexão de um dispositivo embarcado com interface Ethernet a um sistema supervisório através do padrão OPC, bem como sua implementação através de um servidor utilizando a especificação OPC Data Access e estabelecimento de comunicação entre o servidor e o dispositivo utilizando sockets.

O estabelecimento do servidor OPC para o dispositivo, plataforma TINI, possibilitou integrá-lo a um ambiente de automação industrial, sem que seja apenas uma solução específica e dedicada, disponibilizando os dados a qualquer cliente OPC.

Para o desenvolvimento do servidor, utilizou-se a linguagem C++, com o ambiente de desenvolvimento Microsoft Visual C++, para a implementação das interfaces, junto com o conjunto de ferramentas da Softing (Softing OPC Toolbox C++).

Além disso, utilizou-se para o servidor a Interface de Programação de Aplicativos para Microsoft Windows, para o desenvolvimento da comunicação por sockets, a interface da aplicação, bem como funcionalidades de COM/DCOM. A comunicação via sockets para a plataforma TINI foi implementada utilizando a linguagem JAVA, assim como a disponibilização dos dados para envio ao servidor OPC.

Os resultados obtidos foram satisfatórios, uma vez que possibilitaram a comunicação com o dispositivo de maneira eficiente e padronizada, sendo realizados testes do servidor desenvolvido com dois sistemas supervisórios distintos (Matrikon e Wonderware), que também são clientes OPC, demonstrando a utilização do sistema proposto.

Como perspectivas futuras é proposto que seja agregado ao servidor OPC Data Access a implementação do servidor de alarme e eventos e de histórico (OPC-AE e OPC-HDA). Quanto à funcionalidade da solução, pode-se agregar a configuração do endereço IP do dispositivo dinamicamente, além de dispor acesso às portas do controlador.

Referências

COULOURIS, G. F.; DOLLIMORE, J. *Distributed systems : concepts and design*. 4ª. ed. England: Addison-Wesley, 2005.

ISA-95. In: . [s.n.], 1995. Disponível em: <http://www.isa-95.com>. Acesso em: 8 ago. 2007.

IWANITZ, F.; LANGE, J. *OPC - Fundamentals, Implementation and Application*. 3ª. ed. Germany: Hüthing Verlag Heidelberg, 2005.

LING, Z.; CHEN, W.; YU, J. Research and implementation of opc server based on data access specification. *IEEE Fifth World Congress on Intelligent Control and Automation*, v. 2, p. 1475–1478, June 2004.

LIU, J. et al. Using the opc standard for real-time process monitoring and control. *IEEE Software*, v. 22, n. 6, p. 54–59, November 2005.

LOOMIS, D. *The TINI Specification and Developer's Guide*. 1ª. ed. USA: Addison-Wesley, 2001.

MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION. *The Component Object Model Specification Version 0.9*. USA, October 1995.

MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION. *DCOM: Microsoft Distributed Component Object Model Specification Version 1.0*. USA, September 1997.

MICROSOFT CORPORATION AND DIGITAL EQUIPMENT CORPORATION. *MSDN Library Visual Studio 6.0 release*. USA, September 2000.

OPC FOUNDATION. *OPC Data Access Custom Interface Specification Version 3.0*. USA, March 2003.

OPC TASK FORCE. *OPC Overview Version 1.0*. USA, October 1998.

PATTLE, R.; RäMISCH, J. Opc the defacto standard for real time communication. *IEEE Workshop on Parallel and Distributed Real-Time Systems*, p. 289–294, April 1997.

VERYHA, Y. Going beyond performance limitations of opc da implementation. *IEEE Emerging Technologies and Factory Automation*, v. 1, p. 47–50, September 2005.

WONDERWARE CORPORATION. *Intouch: Guia Del Usuario*. USA, 2001.

ZHENG, L.; NAKAGAWA, H. Opc (ole for process control) specification and its developments. *IEEE Society of Instrument and Control Engineers*, p. 917–920, August 2002.