INTEGRAÇÃO DE UMA DESCRIÇÃO DE DISPOSITIVOS ABERTA E NÃO-PROPRIETÁRIA EM SISTEMAS FIELDBUS REAIS E SIMULADOS

Rodrigo Palucci Pantoni*
rodrigoppantoni@yahoo.com.br

Dennis Brandão*
dennis@sel.eesc.usp.br

*Escola de Engenharia de São Carlos - Departamento de Engenharia Elétrica
Universidade de São Paulo
Avenida Trabalhador Sãocarlense, 400 - Centro
São Carlos - SP - Brasil - CEP: 13566-590

RESUMO

Os sistemas de automação industrial contam com a definição de arquiteturas abertas padronizadas, a maioria delas utilizando tecnologias proprietárias. Em sistemas *fieldbus*, a tecnologia normatizada que provê a descrição das características dos dispositivos para o software configurador (IHM - Interface Homem Máquina) é de natureza proprietária e de alto custo, chamada de EDD (*Electronic Device Description*). Alternativamente, a Open-EDD (*Open Electronic Device Description*) é uma tecnologia aberta e não-proprietária baseada em padrões de software. Este trabalho descreve a utilização e validação da Open-EDD em duas diferentes aplicações: num sistema *fieldbus* real e num ambiente *fieldbus* simulado. Os resultados validam a técnica e indicam que esta pode ser aplicada a demais sistemas de automação.

PALAVRAS-CHAVE: Tecnologia aberta, Tecnologia não-proprietária, XML, Device description, FOUNDATION Fieldbus^{TM}, Interoperabilidade, Sistema de automação industrial.

ABSTRACT

Industrial automation systems are based on open standard architectures definitions, in most cases proprietary technologies. In fieldbuses, the standard technology that describes the devices characteristics to the configuration software (HMI) – human machine interface, is proprietary and has high cost, named EDD (Electronic Device Description). Alternatively, the Open-EDD (Open Electronic Device Description) is an open and non-proprietary technology based on software standards. This paper describes the usage and validation of Open-EDD in two different applications: a real fieldbus system and a fieldbus simulation environment. The results validate the proposed technology and indicate that this can be applied to general fieldbus systems.

KEYWORDS: Open technology, Non-proprietary technology, XML, Device description, FOUNDATION Fieldbus^{TM}, Interoperability, Industrial automation system.

1 INTRODUÇÃO

O conceito de interoperabilidade é a habilidade de um sistema ou de um produto de trabalhar com outros sistemas ou produtos sem exigir esforços para integração (Miller, 2002). Outro conceito ligado é o conceito de intercambiabilidade, que define a capacidade de troca de um sistema ou produto sem a perda de funcionalidade num contexto mais amplo (Donaires, 2004).

Pode-se definir um software aberto como um software que possui uma arquitetura que foi especialmente projetada para prever integração com outras tecnologias, um requisito não-funcional do projeto de software.

Um outro aspecto envolvido é o conceito de padrão aberto, que pode ser implementado livremente sem restrições legais

Artigo submetido em 03/08/2007 (Id: 815)
Revisado em 29/09/2008 e em 03/12/2008
Aceito sob recomendação do Editor Associado Prof. José Reinaldo Silva

ou comerciais, isto é, se for necessário o pagamento de royalties então o software não é aberto (no sentido de livre). A questão de propriedade não está vinculada estritamente ao custo, mas sim aos direitos sobre o objeto (tecnologia no caso) de forma mais ampla. Na prática, o que carateriza uma tecnologia como proprietária ou não-proprietária é a licença de uso vinculada à tecnologia pelo detentor dos direitos intelectuais sobre a mesma. Ou a tecnologia é de domínio público, e nesse caso é inerentemente não-proprietária, ou é de domínio de uma pessoa física ou jurídica que detém os direitos sobre a mesma. Nesse segundo caso, esse detentor tem a premissa legal de associar uma licença de uso para sua propriedade. Essa licença pode ser restritiva, no sentido de que confere ao usuário um subconjunto menor de direitos em relação àqueles do proprietário. Licenças restritivas são inerentemente discriminatórias ao impor diferentes liberdades de uso a diferentes usuários, restringindo, por exemplo, quem, como, quando ou onde a tecnologia pode ser utilizada. Por outro lado, se a licença é não-restritiva, está garantida a igualdade de direitos de uso, de onde surge o termo "livre", que é utilizado, por exemplo, em "*Free Software*". O termo "aberto" também é utilizado como em "*Open Source*" (Monaco, 2006).

Neste trabalho o termo "aberto" é usado em relação à arquitetura projetada para prever integração, enquanto que o termo "não-proprietário" é usado no sentido de liberdade de uso, em relação à licença de uso envolvida.

A tecnologia EDD (*Electronic Device Description*) é utilizada para descrever as características de dispositivos de campo utilizados em automação industrial *fieldbus* para os sistemas configuradores. A EDD é a linguagem comum que todos os fabricantes usam para descrever seus dispositivos. Tal tecnologia permite a interoperabilidade entre os dispositivos no nível do configurador *fieldbus*. Atualmente essa tecnologia é aplicada para os dispositivos dos protocolos FOUNDATION Fieldbus^{TM} (FF), HART e PROFIBUS (Zielinsky, 2005).

A tecnologia aberta e não proprietária proposta neste trabalho, chamada de Open-EDD (Pantoni et al., 2007) diferencia-se das atuais tecnologias de descrição de dispositivos: é baseada em padrões abertos já vastamente empregados na área de tecnologia de informação, porém adaptados para área industrial. Neste estudo, a tecnologia base aberta e não-proprietária é a chamada XML (*eXtensible Markup Language*), que é mundiamente conhecida pela integração de aplicações de Internet. Essa tecnologia é prosposta para ser uma alternativa tecnologica para a EDD, que é definida e controlada pela Fieldbus Foundation^{TM}, para o protocolo FOUNDATION Fieldbus^{TM} (FF).

A motivação para o desenvolvimento da Open-EDD é apresentada na próxima seção; a seção 3 descreve a tecnologia EDD e sua utilização; a seção 4 aborda os trabalhos relacionados encontrados na literatura; a seção 5 detalha o software configurador utilizado para validar a tecnologia proposta (no sistema *fieldbus* real); a seção 6 descreve as características da Open-EDD; a seção 7 descreve a metodologia de integração num sistema *fieldbus* real; a seção 8 descreve a metodologia de integração num sistema *fieldbus* simulado e suas características, e por fim, a última seção delinea as conclusões.

2 MOTIVAÇÃO DO TRABALHO

A tecnologia EDD para FF consiste em uma linguagem padronizada, a IEC 61804-2 EDDL (*Electronic Device Description Language*), um compilador de EDD e um interpretador de informações, chamado de DDS (*Device Description Services*).

O compilador traduz o código fonte da descrição criada utilizando a EDDL, e gera um arquivo binário especial, codificado e compactado. O interpretador possui serviços para extração de informações desse arquivo binário. Além disso, ele deve obrigatoriamente ser integrado no configurador FF ou qualquer outra aplicação de interação homem máquina (IHM) que tenha que interagir com os dispositivos de campo.

Embora a EDDL seja um padrão IEC, o compilador e o interpretador são proprietários (Yong et al, 2004) Tais elementos são distribuídos e comercializados pela FOUNDATION Fieldbus^{TM}.

EDD pode ser considerada uma tecnologia madura, que acumula recursos, funcionalidades e melhorias durante mais de 12 anos de revisões técnicas. Entretanto, para que seja desenvolvido um dispositivo ou qualquer software que interaja com os dispositivos, é necessária a aquisição dos chamados *toolkits* de desenvolvimento (compilador e interpretador). Tal fato exclue potencialmente universidades e centros de pesquisa em razão do alto custo de desenvolvimento. O trabalho aqui proposto então contribui para o desenvolvimento como, por exemplo, de simuladores de rede, de blocos funcionais e aplicações de sistemas de automação.

A Open-EDD também simplifica o trabalho de desenvolvimento para o desenvolvedor de dispositivos tanto como para o desenvolvedor de configuradores (IHM). Para desenvolvedores de dispositivos, a EDD atualmente usa a EDDL, similar à linguagem C, que requer treinamento e tempo para aquirir conhecimento. Para IHM, a integração do DDS é também árdua porque deve ser feita também em linguagem C.

A criação da Open-EDD envolve a definição da nova linguagem chamada de Open-EDDML (*Open Electronic Device Description Markup Language*), projeto e implementação do

compilador e do interpretador (análogo a EDD).

A linguagem Open-EDDML, em sua versão inicial, inclui a informação das estruturas de identificação, de listas de blocos e parâmetros de um dispositivos. Todas essas informações são definidas na especificação da EDD definida pela FOUNDATION Fieldbus^{TM} (Fieldbus Foundation , 1999a).

3 EDD

Para que o software configurador possa realizar configurações nos dispositivos, configurar as redes *fieldbus* e programar a lógica da estratégia de controle, é necessário que o próprio saiba interagir com os dispositivos. Para isso, o software necessita ter informações sobre as características dos equipamentos, tais como nome e tipo das variáveis, funções das variáveis, as entradas e saídas, entre outros.

No caso do FF, a descrição das características dos dispositivos é disponibilizada para o configurador através da tecnologia EDD. O software configurador tem um banco de EDD's disponíveis em seu domínio, que compõe todos os dispositivos suportados pelo configurador, uma EDD para cada dispositivo. Conseqüentemente, a integração de um novo dispositivo no sistema no nível do configurador é feito integrando-se a EDD do mesmo.

Além da descrição, o principal papel da EDD é possibilitar a interoperabilidade de dispositivos ou equipamentos de diferentes fabricantes nos diferentes sistemas para automação de também diferentes fabricantes.

A EDD também é conhecida por DD (*Device Description*), que é o nome dado à versão antecessora da EDD. Por essa razão, neste trabalho, as siglas DD e EDD expressam a mesma tecnologia, com a diferença de versão.

Uma das primeiras publicações da DD foi na Alemanha. Na época, a mesma foi apresentada como a oitava camada do modelo de referência ISO/OSI (Borst, 1991). A maioria dos especialistas da área não entendeu ou discordou dessa definição pelo fato do modelo possuir apenas sete camadas. Na arquitetura do protocolo FF, a EDD (ou DD) localiza-se na chamada camada do usuário, junto com os chamados blocos funcionais, pois é esta que descreve as informações dos blocos para um nível mais alto, o do software configurador, por exemplo. A figura 1 mostra o modelo de camadas FF e localização da camada do usuário, que é acima da sétima camada de referência do modelo ISO/OSI.

Outras tecnologias de descrição foram criadas para garantir a interoperabilidade em outros protocolos de comunicação industrial: PROFIBUS introduziu GSD (*General Station Description*); PROFINET introduziu GSDML (*General Station Description Markup Language*) e INTERBUS introduziu FDCML (*Field Device Configuration Markup Language*).

Além da EDD, os protocolos FF, HART e PROFIBUS também utilizam uma tecnologia alternativa (menos utilizada) considerada como complementar a EDD (Kastner et al, 2004) chamada de FDT/DTM (*Field Device Tool/Device Type Manager*) (FDT Group, 2006).

No protocolo FF, as informações da EDD podem ser utilizadas pelo configurador em duas situações: coleta de informações das características dos dispositivos para configuração *offline* e para comunicação com os dispositivos (*online*) para configurações diretas nos dispositivos.

A configuração no modo *offline* foi possibilitada com o advento das tecnologias de descrição de dispositivos, por descrever as características dos dispositivos integralmente, como se estes estivessem conectados no sistema. Tal modo permite que a iniciação do trabalho de engenharia seja num local diferente do da planta, onde efetivamente ocorrerá o controle do processo.

O modo de configuração *online* também utiliza as informações da EDD para realizar configurações diretas com os dispositivos, funcionado como *drivers*. Tais configurações diretas podem ser, por exemplo: envio de comandos de escrita ou leitura para os parâmetros dos blocos dos dispositivos, instanciação de blocos funcionais nos dispositivos, descarregamento da estratégia de controle, etc. No modo *online*, o configurador necessita acessar os dispositivos da rede, e para isso utiliza o componente servidor OPC – *OLE for Process Control* (OPC Foundation, 2006). Tal componente necessita de descrições dos dispositivos para acessar os dispositivos; para isso ele requisita tais informações para o chamado servidor de DD (figura 2).

A figura 2 mostra a arquitetura projetada para o configurador Syscon (Smar Equipamentos Industriais, 2005). Nessa versão, foi utilizada a DD de versão 4 (Fieldbus Foundation, 1999a).

O banco de DD's é armazenado no disco rígido da estação de engenharia, através de uma estrutura de diretórios (chamado em sistemas *fieldbus* comumente de *Device Support*), cada um representando um tipo de dispositivo suportado. Dentro desses diretórios, para cada versão de dispositivo, existe um arquivo proprietário binário (extensão ffo), um arquivo com a lista de símbolos do arquivo binário correspondente ao arquivo binário (extensão sym) e um arquivo CF (*Capabilities File*, de extensão cff).

A descrição em arquivos CF é também utilizada em dispositivos FF, porém com um objetivo diferente da EDD: é utilizada para descrever as capacidades físicas do dispositivo,

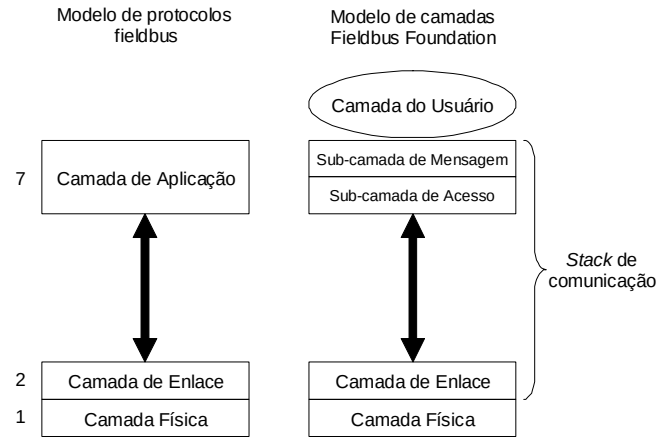

Figura 1: Camada do usuário FF (Smar Equipamentos Industriais, 2005).

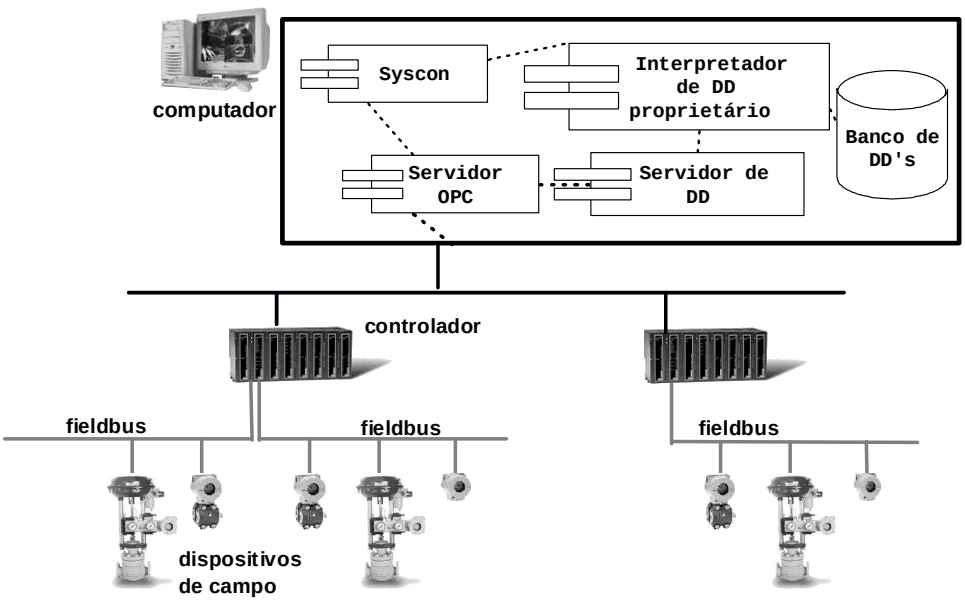

Figura 2: Arquitetura do software Syscon (Pantoni, 2006).

como o máximo de memória disponível para cada recurso, quantidade de ligações entre os blocos, entre outras (Fieldbus Foundation, 1999b). CF é um arquivo de formato multiplataforma ASCII (*American Standard Code for Information Interchange*). O arquivo CF pode ser criado usando qualquer editor de textos como, por exemplo, o bloco de notas do sistema operacional Windows. O formato do arquivo tem que ser orientado a linhas; cada linha deve conter uma expressão relativa à cada informação do dispositivo.

4 TRABALHOS RELACIONADOS

Esta seção tem como objetivo abordar uma breve revisão dos trabalhos relacionados encontrados na literatura de estudos envolvendo formatos de descrição de dispositivos.

Metodologias para integrar EDDL e FDT/DTM foram estudadas nesta década (Simon et al, 2003) (Kastner e Kastner-Masilko, 2004). A desvantagem dessa técnica é que o FDT/DTM é baseado em COM (*Component Object Model*) (Microsoft, 2008), que é uma tecnologia proprietária desenvolvida pela Microsoft e só é suportada pelo sistema operacional Windows. Pior que isso, Merrit (2002) afirma que a tecnologia FDT/DTM nem sobrevive a outras gerações dos próximos sistemas operacionais Windows, porque todos os componentes de device (chamados de DTM's) teriam que ser reescritos pelo fato desses futuros sistemas operacionais não suportarem mais COM.

Dessa forma, a tecnologia resultante desta combinação não pode ser considerável totalmente interoperável com todos os sistemas de automação. Além disso, FDT/DTM é relativamente difícil de ser desenvolvida e não suporta persistência de dados (Kastner e Kastner-Masilko, 2004). Por outro lado, EDDL suporta persistência de dados (Zielinsky, 2005).

Outro fato que merece menção é a desvantagem da tecnologia FTD/DTM ser um componente de software que contém todas as mensagens e telas ("*look and feel*") da interação com o usuário. A liberdade para implementação de novos recursos pode parecer uma vantagem à primeira instância, mas na realidade pode ser uma outra desvantagem. Para o funcionamento dos recursos, existem interfaces definidas pelas especificações (com o objetivo da padronização); assim a implementação de novos recursos seria não-padrão, já que não existem regras para incorporação no sistema de automação, e assim no configurador, em razão da ausência de interfaces padronizadas. Por outro lado, implementações de novos recursos agregariam mais valor a determinado produto, no caso o do fabricante que tiver maior competência e criatividade.

O processo de desenvolvimento de um DTM necessita de um compilador, igualmente ao processo de uma DD. Como o DTM é baseado em um componente COM, o compilador pode estar em qualquer ambiente de desenvolvimento de software que suporte o desenvolvimento da tecnologia COM. Pode-se citar como exemplo desses compiladores: Borland Delphi, Microsoft Visual Basic, Microsoft Visual C++, Borland C++, etc. Esses compiladores são comercializados assim como o compilador da tecnologia EDDL, embora nesse caso haja liberdade de escolha. Assim, o processo de desenvolvimento do DTM gera o mesmo problema da tecnologia EDD: a aquisição de um compilador proprietário.

Na análise apresentada por (Wollschlaeger, 1999), é proposto uma descrição de dispositivos para CANOpen baseada em XML. Nesse estudo o autor apresenta o modelo de software desenvolvido e expõe os benefícios amplamente conhecidos da utilização da tecnologia XML.

Em relação a trabalhos de descrição de dispositivos envolvendo os protocolos Fieldbus FF, HART e PROFIBUS, um trabalho similar ao proposto neste trabalho é abordado em (Wang et al, 2004), o qual os autores analisam analiticamente a possibilidade de mapear os dados da EDD em XML, não abordando assim uma implementação e validação da metodologia. Esse estudo não previa tipos de dados complexos das variáveis dos dispositivos tais como *Records* que contém informações *Enumerated* e *BitEnumerated*, as quais a manipulação é árdua computacionamente (mencionadas na tabela 1).

Além da utilização de XML Schema para validação, este trabalho difere da prosposta apresentada por (Wang et al, 2004) nos seguintes aspectos: neste trabalho a implementação e validação da tecnologia é efetivamente realizada; foi desenvolvido um modelo de software com as construções da EDD para manter a compatibilidade com o padrão EDDL; a tecnologia proposta é baseada em padrões abertos e não-proprietários de software; a utilização de características herdadas da XML tais como geração automática e padrão de uma da interface gráfica através da combinação da XSLT (*eXtensible Stylesheet Language Transformation*).

5 CONFIGURADOR *FIELDBUS*

Para efeito de entendimento, torna-se necessário um detalhamento do aplicativo configurador utilizado para a validação da Open-EDD num sistema *fieldbus* real. Este configurador, chamado de Syscon, é utilizado para realizar configurações em dispositivos FF, programando-se assim estratégias de controle de processos *fieldbus* reais. Syscon foi desenvolvido na linguagem C++, no ambiente Borland C++ 5.02.

A Fieldbus Foundation^{TM}, que administra o protocolo FF, especifica para programação da estratégia de controle uma linguagem de programação gráfica simples. Essa programação é composta basicamente por diagramas de blocos funcionais ligados entre si (*links*). Tais blocos são programas residentes nos dispositivos da rede e encapsulam funções e algoritmos básicos de automação e controle de processos.

A configuração distribui os blocos funcionais em diferentes equipamentos de campo, caracterizando assim uma estratégia de controle distribuída como mostra a Figura 3. Exemplificando, o blocos "TT1-AI-1" e "TT1-PID-1" são de um transmissor de temperatura e o bloco "FI1-AO-1" é de um posicionador de válvulas.

As operações no Syscon envolvem principalmente configurações em blocos funcionais. As operações básicas e essenciais de configuração do Syscon são: configuração *offline* de estratégia de controle, construção da topologia de rede dos dispositivos, configuração *offline* de valores dos parâmetros dos blocos, descarregamento dessas configurações para as redes e dispositivos (*download*) e configuração *online* de valores dos parâmetros dos blocos.

A estratégia de controle é configurada no modo *offline* utilizando diagramas gráficos para a realização de ligações entre os blocos funcionais (Fieldbus Foundation, 1999c) (Fieldbus Foundation, 1999d) (figura 3).

A topologia de dispositivos é construída se adicionando redes ou canais *fieldbus* e se configurando valores de tempo de ciclo de comunicação (chamado de macrociclo no protocolo FF) para cada um deles. Em cada *fieldbus*, são adicionados dispositivos. Nesses dispositivos, por sua vez são criadas

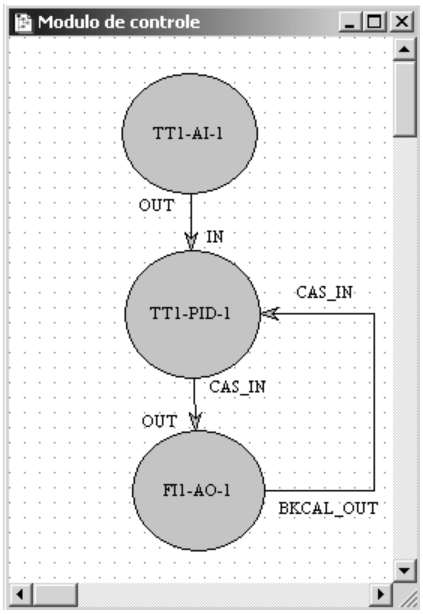

Figura 3: Estratégia de controle desenvolvida no modo offline.

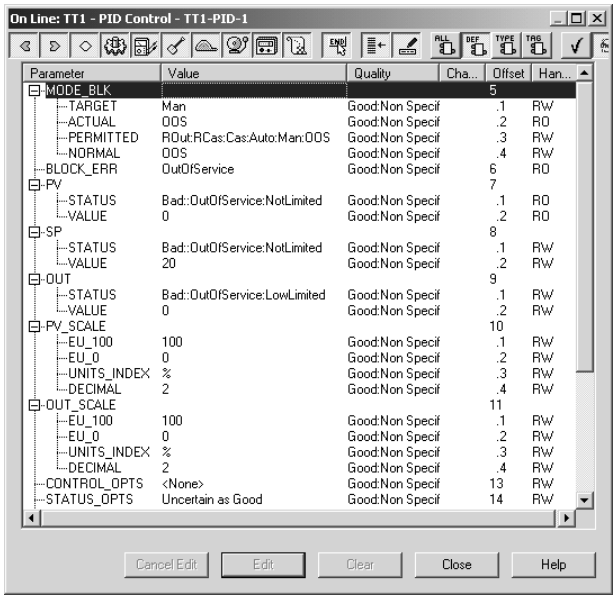

Figura 5: Janela de mudança de valores online.

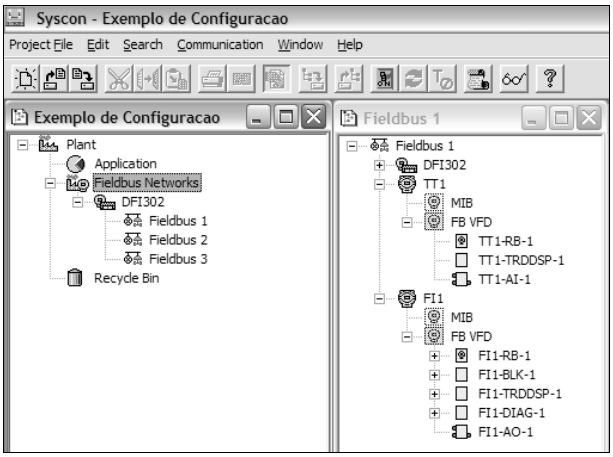

Figura 4: Topologia de rede no configurador.

instâncias de blocos funcionais. A figura 4 mostra a tela do configurador com uma topologia exemplo criada.

A configuração dos dispositivos no modo *offline* é feita para que posteriormente seja descarregada nos dispositivos e nas redes através da operação de *download*.

A configuração no modo *online* consiste em editar os valores dos blocos funcionais dos dispositivos já em operação. A figura 5 mostra a janela de mudança de valores *online*.

A tabela 1 mostra a classificação dos valores contidos na EDD de um dispositivo.

6 OPEN-EDD

6.1 Linguagem de programação

Análogo a EDDL IEC 61804-2- a linguagem de programação que define a EDD – uma linguagem baseada em XML foi criada para desenvolver arquivos de Open-EDD. Essa linguagem é chamada de Open-EDDML. Assim, a EDDL e a Open-EDDML possuem o mesmo conteúdo, porém em diferentes formatos.

Para criar uma linguagem baseada em XML é necessário descrever um conjunto de regras que definem os elementos ou marcadores XML (Thiruvathukal, 2004). Existem duas possibilidades para definição dessa linguagem: DTD (*Document Type Definition*) ou XML Schema (XML Schema, 2006) (Walmsley, 2002). Este trabalho utilizou o XML Schema em razão da necessidade de descrição de tipos complexos de variáveis dos dispositivos e pelo mapeamento natural com UML, características que o DTD não oferece. O DTD, que foi avaliado para o mesmo propósito por (Wang et al, 2004), não foi adotado neste trabalho por não suportar tipos complexos e não possuir associação intuitiva com a notação UML, já que é natural em desenvolvimento de software associar uma classe a um elemento XML.

A tabela 2 mostra uma comparação das linguagens de EDDL e Open-EDDML.

Baseado no XML Schema desenvolvido, um diagrama UML foi criado para representar as associações entre as classes que descrevem os elementos da Open-EDDML, tais como *De-*

Tabela 1: Classificação de valores da EDD

Informação	Descrição	Valores Possíveis
Handling	Indica se o parâmetro é para leitura, escrita ou ambos. É configurado para permitir edição de valores.	*Read, Write, Read/Write.*
Class	Define a classe do parâmetro. A classe dá uma referência para saber se o parâmetro é se entrada ou saída, por exemplo, para criação de estratégias de controle.	*Input, Output, Contained, Dynamic, Diagnostic, Service, Operate, Alarm, Tune, Local.*
Type	Define o tipo do parâmetro, indica se o formato do dado que será mostrado no aplicativo e o tipo de dado para ser enviado para o dispositivo.	*Ascii, BitString, BitEnumerated, BooleanT, DateAndTime, Double, Duration, Enumerated, Euc, Float, Index, Integer, OctetString, PackedAscii, Password, Time, Time_Value, Unsigned, VisibleString.*
Metatype	Define o *Metatype*.	*Array, Variable, Record*.
BitEnumeration	Define a lista para de elementos caso haja bit-enumerações.	Específico para cada parâmetro.
Enumeration	Define a lista para de elementos caso haja enumerações.	Específico para cada parâmetro.

viceDescriptionXMLELEM, *BlockXMLElem*, entre outros, como indicado na figura 6.

6.2 Interface gráfica amigável para os dados da Open-EDD

Uma característica inerente a tecnologias derivadas da tecnologia XML é o fato de poder unir-se com uma gama de tecnologias de internet tais que possam lidar com linguagens tais como HTML, Javascript, entre outras de forma natural.

Dessa forma, a Open-EDD tem um potencial de gerar interface padronizada de forma automática. Para cada arquivo de Open-EDD, é gerado a mesma interface gráfica em termos de estilo e "*look and feel*", mas com diferente informações.

A criação do mecanismo de geração automática de uma interface amigável utilizando os dados da Open-EDD foi executada implementando-se uma lógica XSLT (*eXtensible Sty-*

Tabela 2: Comparação entre as linguagens EDDL e Open-EDDML

EDDL	Open-EDDML
MANUFACTURER 1, DEVICE_TYPE 1, DEVICE_REVISION 1, DD_REVISION 1	<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
BLOCK block_1	<?xml:stylesheet type='text/xsl' href='OpenEDD.xsl' ?>
{LABEL "Example block"; PARAMETERS	<DeviceDescription>
{param_1, float_var, "float_parameter", "This is a float parameter for block_1"; param_2, record_of_vars, "This is a record parameter for block_1";}	<Identification DDRevision="0x1" DeviceRevision="0x1" DeviceTypeId="0x1" DeviceTypeName="Hipotetic Device" ManufacturerId="0x1" ManufacturerName= "Hipotetic Manufacturer" />
}	
VARIABLE float_var	<BlockList>
{LABEL "float"; CLASS OUTPUT; TYPE FLOAT; { DISPLAY_FORMAT "6.6f" EDIT_FORMAT "3.3f"} HANDLING READ & WRITE;	<Block Id="0x1" Name="block_1" Label="Example block">
}	<ParameterList NumberOfParameters="2">
RECORD record_of_vars	<Parameter Handling="0x3" DisplayFormat="6.6f" EditFormat="3.3f" Id="0x1" MetaType="S" Name="float_var" Label="float variable" ParameterClass="0x004000" Size="4" Type="5" />
{LABEL "Record of variables"; MEMBERS {member_1, integer_var, "integer variable"; member_2, ascii_var, "ascii variable"; }	<Parameter Id="0x80020126" MetaType="R" Name="record_of_vars" ParameterClass="0x8000">
}	<MemberList NumberOfElements="2">
VARIABLE ascii_var	<Member DisplayFormat="u" EditFormat="u" Handling="0x1" Id="0x1" MetaType="S" Name="integer_var" Label="integer variable" ParameterClass="0x8000" Size="1" Type="2"/>
{ LABEL "ASCII variable"; TYPE ASCII (10); CLASS CONTAINED; HANDLING READ & WRITE;}	<Member Handling="0x3" Id="0x80020020" MetaType="S" Name="ascii_var" Label="ascii variable" ParameterClass="0x8000" Size="10" Type="9" />
VARIABLE integer_var	</MemberList>
{LABEL "integer variable"; TYPE INTEGER; CLASS CONTAINED; HANDLING READ;}	</Parameter>
	</ParameterList>
	</Block>
	</BlockList>
	</DeviceDescription>

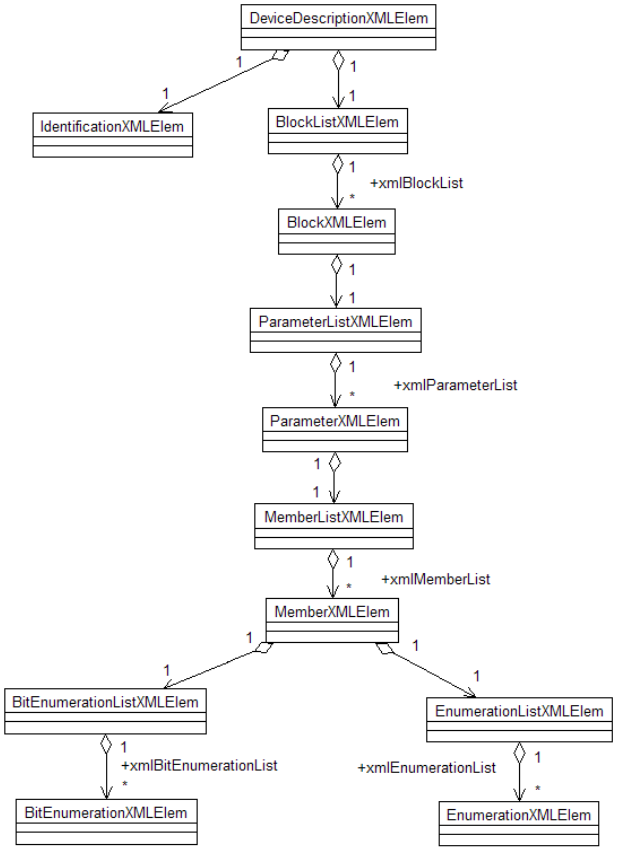

Figura 6: Associação dos elementos da Open-EDDML na notação UML.

lesheet Language Transformation), que é uma tecnologia muito utilizada em navegadores de internet. O produto do processo é uma tela HTML. A lógica XSLT é descrita no arquivo "OpenEDD.xsl", que é associado a qualquer arquivo Open-EDD para geração de interface gráfica. Complementarmente ao arquivo "OpenEDD.xsl", um arquivo de estilo chamado de "OpenEDD.css" foi criado para armazenar o estilo do HTML, incluindo tamanho de fontes e tipo, cores, entre outros.

A figura 7 mostra a visão gerada através do mecanismo de geração automática utilizando uma Open-EDD de um dispositivo do fabricante Endress+Hauser.

7 INTEGRAÇÃO NO SISTEMA FIELDBUS REAL

7.1 Compilador e interpretador

No caso da integração no sistema *fieldbus* real, um compilador e um interpretador foram desenvolvidos para leitura de informações no formato XML.

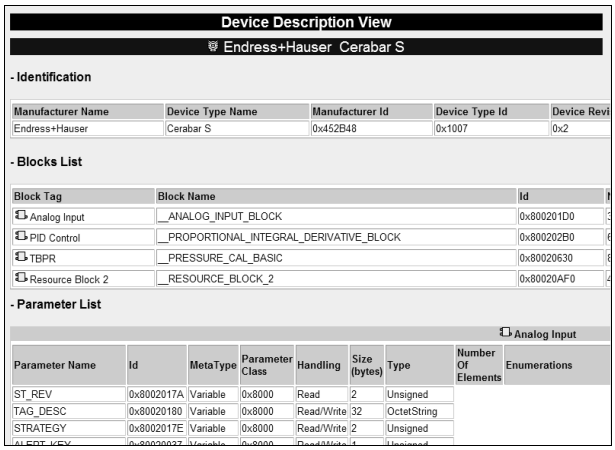

Figura 7: Interface gráfica amigável.

Analogamente ao compilador e ao interpretador da tecnologia EDD, um único componente de software foi desenvolvido para compilar e interpretar informações da Open-EDD.

Esse componente (figura 8) foi implementado utilizando o ambiente Borland C++ 5.02 e a ferramenta de modelagem UML Rational Rose 98. Uma biblioteca de código aberto e livre chamada de Xerces (Apache Software Foundation, 2005) foi acoplada no componente desenvolvido para executar operações básicas de escrita e leitura de dados em XML.

Neste trabalho foi utilizada a API DOM (Document Object Model) implementada em Xerces, para manipulação dos dados em XML. Essa tecnologia foi escolhida por ser um pa-

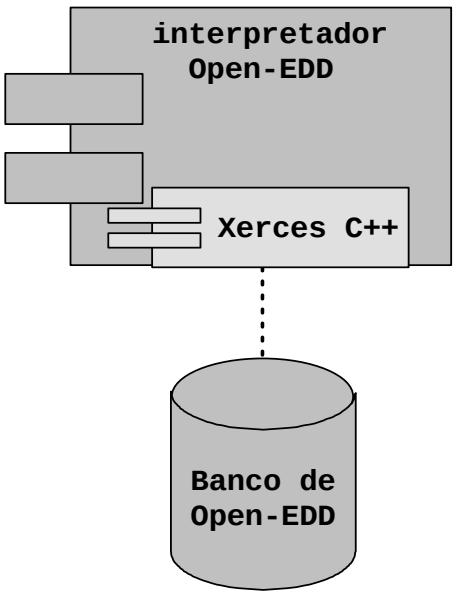

Figura 8: Componente e a biblioteca Xerces.

drão W3C, por prover mecanismos de validação automáticos juntamente com XML Schema (se comparado a EDD, esse mecanismo atua como um compilador para validação), e por representar os marcadores ou elementos como objetos (abstração e encapsulamento) (XML DOM, 2007). Tais vantagens, por exemplo, não são encontradas na API SAX (School Of Computing And IT, 2007).

Compilação dos dados foi herdada da biblioteca Xerces C++, que possui serviços de validação do XML quando se tem um XML Schema como referência. Para cada dispositivo que for adicionado no configurador, um serviço de validação é disparado e assim o arquivo de Open-EDD desse dispositivo é validado.

Modelagem em objetos foi utilizada para desenvolvimento do componente de software. O desenvolvimento de sistemas complexos e de alto nível é melhor gerenciado e modelado utilizando-se orientação a objetos (OO) e UML (Booch, 1994) (Booch. et al, 1999).

Dois diagramas de classes foram criados para modelar a estrutura do XML Schema. No primeiro diagrama desenvolvido, uma classe chamada de *XMLElement* foi criada. Essa classe possui serviços básicos de leitura e escrita e de validação de informações, providos pela biblioteca Xerces. Cada elemento ou marcador XML da Open-EDD é herdado dessa classe de serviços básicos. O segundo diagrama representa as relações de associação entre os marcadores XML, a mesma informação de associação do XML Schema (figura 6).

A metodologia para integração do componente de Open-EDD no configurador foi executada substituindo-se o componente proprietário DDS pelo desenvolvido neste trabalho. Conseqüentemente, o configurador e o servidor de DD tiveram que ser adaptados, como mostrado na figura 9.

No banco de DD's, arquivos de extensão ffo e sym (gerados pelo compilador proprietário) foram substituídos por um único arquivo de extensão XML (Open-EDD) em cada diretório da estrutura *Device Support*. Na figura 9, é representado como Banco de DD's Open EDDML.

7.2 Procedimento de testes

Os seguintes passos foram executados para validação da tecnologia proposta:

- Arquivos de Open-EDD foram criados baseados em dispositivos devidamente certificados do fabricante Smar (DFI302, TT302 e FY302) utilizando o editor de XML livre Netbeans, disponível em Netbeans (2006).
- Os arquivos criados foram copiados no banco de Open-EDD;
- Os dispositivos foram criados na configuração do Syscon;
- Os blocos *Resource* e *Transducer* de todos os dispositivos foram criados na configuração do Syscon. Para o TT302, os blocos *Analog Input* (AI) e PID também foram criados. Para o FY302, o bloco *Analog Output* (AO) foi criado.
- Uma estratégia de controle AI-PID-AO foi criada (igualmente a mostrada na figura 3);
- Os blocos foram parametrizados no modo *offline*;
- A configuração foi descarregada nos dispositivos (*download*);
- Valores *online* foram lidos e escritos nos dispositivos de campo.

8 INTEGRAÇÃO NO SISTEMA FIELDBUS SIMULADO

A integração num ambiente simulado foi feito no simulador de protocolo FOUNDATION Fieldbus^{TM} denominado FBSIMU (Pinotti Jr e Brandão, 2005) (Pantoni et al., 2008).

O simulador foi desenvolvido em LabVIEW, utilizando a linguagem de programação G, nativa do ambiente. Cada módulo do FBSIMU simula uma estrutura de um sistema FF real, tais como escalonamento de mensagens na rede, algoritmos de controle, blocos funcionais, processos controlados, processos de supervisão entre outros.

Dentre as funcionalidades do ambiente FBSIMU, pode-se destacar:

- Proporcionar ferramentas e *frameworks* para o projeto e implementação de blocos funcionais customizados ou blocos funcionais padrão no ambiente de simulação;
- Executar blocos funcionais de forma contínua ou única, com a possibilidade de monitorar e modificar parâmetros de configuração e de entrada;
- Configurar e executar malhas de controle de blocos funcionais segundo uma tabela configurável de escalonamento de blocos funcionais e de mensagens;
- Utilizar modelos de simulação de sistemas dinâmicos, conjugados às malhas de blocos funcionais, de modo a estabelecer um sistema de controle de malha fechada sincronizado com a planta simulada;
- Realizar a aquisição de dados de barramento de *fieldbuses* reais e analisá-los de modo a estimar parâmetros e variáveis de comunicação.

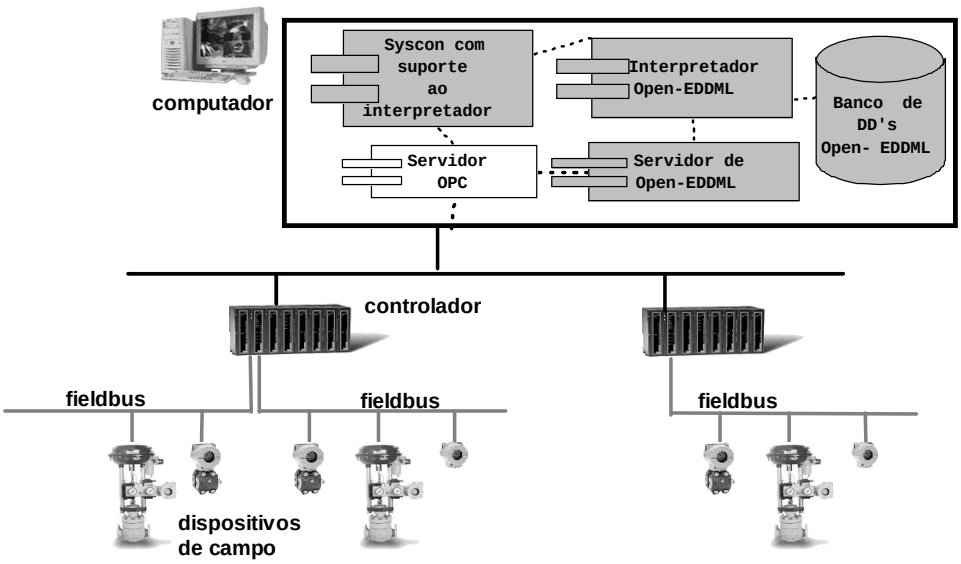

Figura 9: Arquitetura do software Syscon modificada.

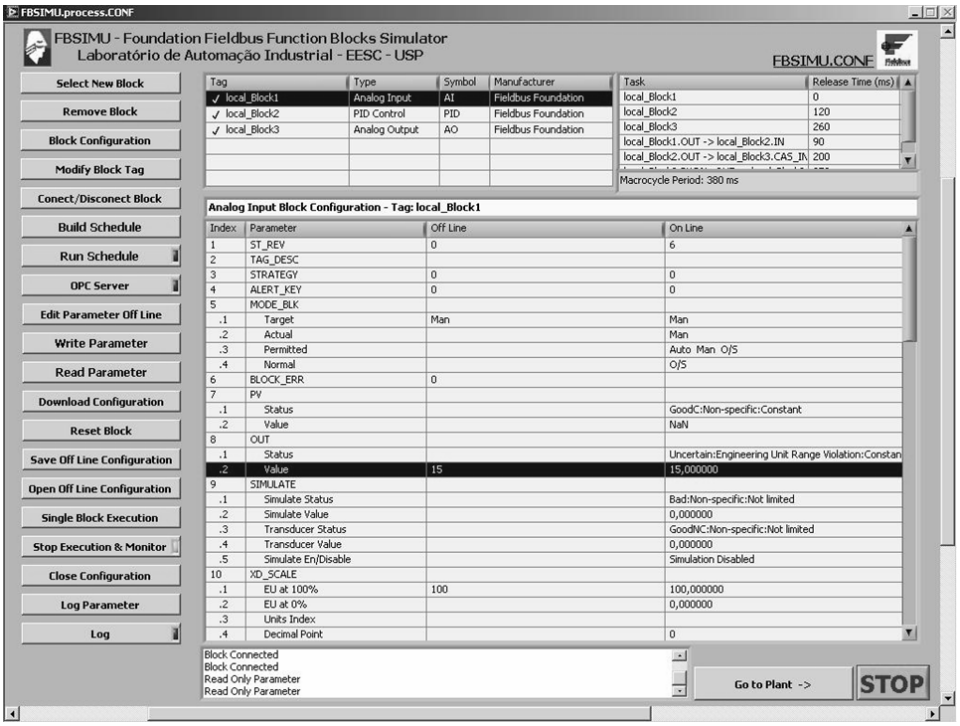

Figura 10: Tela de configuração de blocos e parâmetros do FBSIMU.

Os blocos funcionais e parâmetros são implementados em código no simulador, isto é, para cada dispositivo a ser inserido na simulação, é preciso que se implemente seus blocos, lista de parâmetros com seus respectivos atributos tais como *metatype*, *class*, valores possíveis, entre outros (apresentados na tabela 1).

Por outro lado, a integração de uma *Device Description* no simulador permite o desenvolvimento de um algoritmo para generalização da metodologia para supervisão e configuração de forma automática no sistema. Para isso, seria apenas necessário integrar no simulador o arquivo de DD fornecido pelo fabricante relativo ao dispositivo em questão.

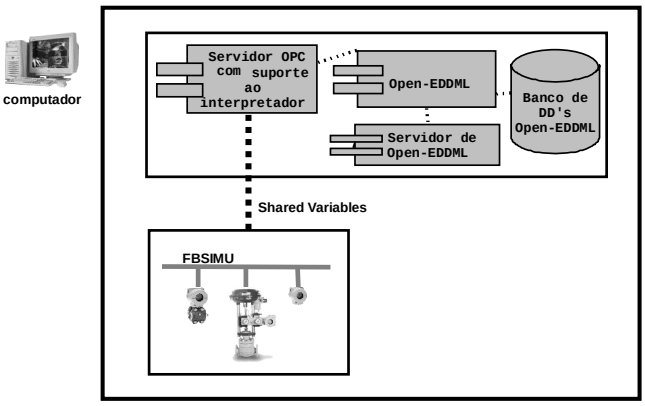

Figura 11: Arquitetura do software FBSIMU integrado ao servidor OPC & Open-EDD.

Dessa maneira, a Open-EDD aparece como uma alternativa para viabilizar esta característica desejada no simulador.

Foi desenvolvido um aplicativo conversor de EDD para Open-EDD o qual converte de forma automática os arquivos binários para o formato aberto e não-proprietário XML. O conversor utiliza o mesmo componente de software desenvolvido apresentado na figura 8, o qual utiliza a biblioteca Xerces para geração de XML. Para leitura de arquivos de DD codificados, foi acoplado do interpretador DDS. Portanto, na conversão, gera-se um aquivo de Open-DD correspondente a um arquivo de DD em formato binário.

Assim, para cada dispositivo a ser integrado no simulador, basta ter em mãos o arquivo de Open-DD. Uma vez que a programação de cada bloco funcional segue as especificações e descrições de tal bloco no arquivo de Open-EDD, tal arquivo é utilizado para a interpretação dos dados para supervisão de estratégias de controle no simulador.

A supervisão da simulação é realizada por meio de uma interface OPC. O servidor OPC utilizado neste projeto foi desenvolvido com capacidade de interface para "*Shared Variables*", tecnologia da National Instruments para o intercâmbio de dados com aplicações LabVIEW (National Instruments, 2007). Na aplicação proposta, as *Shared Variables* são transmitidas em formato de *string* textual do simulador para o servidor OPC, conforme a figura 11.

A integração dos arquivos Open-EDD no servidor OPC do simulador é efetivada através da inclusão destes em um diretório denominado *Device Support*, interpretados pelo módulo servidor para a atribuição dos tipos de dados de cada variável monitorada (*data casting*) através do atributo *metatype* correspondente.

8.1 Procedimento de testes

Os seguintes passos foram executados para validação funcional da tecnologia proposta no simulador:

- Arquivos de EDD de dispositivos devidamente certificados do fabricante Smar (DFI302, TT302 e FY302) foram convertidos para Open-EDD utilizando o software conversor. Os arquivos criados foram copiados no banco de Open-EDD;
- Os dispositivos foram criados na configuração do Simulador;
- Para o TT302, os blocos *Analog Input* (AI) e PID também foram criados. Para o FY302, o bloco *Analog Output* (AO) foi criado.
- Uma estratégia de controle AI-PID-AO foi criada (igualmente a mostrada na figura 3);
- Os blocos foram parametrizados no modo *offline*;
- Valores *online* foram lidos nos dispositivos de campo simulados, através da interface de supervisão OPC. Na figura 12 apresenta a interface de um cliente OPC a monitorar parâmetros do bloco PID simulado.

9 CONSIDERAÇÕES EXPERIMENTAIS

Uma desvantagem da Open-EDD conhecida é que os arquivos podem ter um tamanho significativo no disco rígido dependendo do número de informações incluídas (o tamanho

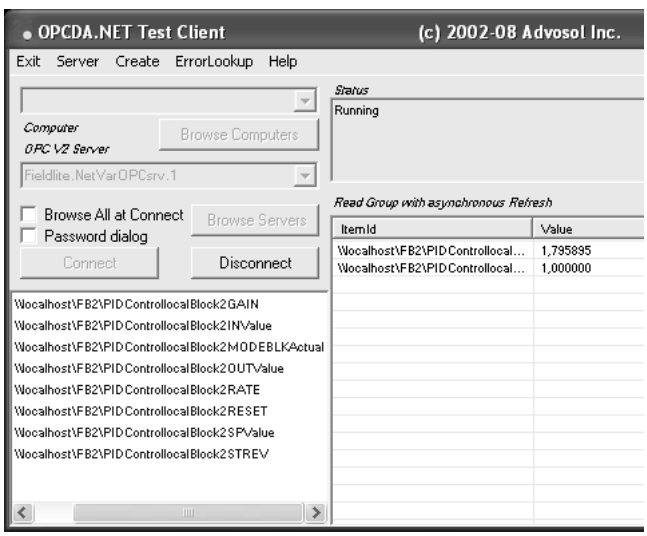

Figura 12: Monitoramento online de parâmetros com o uso das Open-EDDs.

varia entre 10 Kbytes and 2 Megabytes). Arquivos EDD são menores, pois são arquivos compactados e binários, enquanto que arquivos Open-EDD são ASCII (XML). Em razão do tamanho que os arquivos de Open-EDD podem alcançar, a evolução desta tecnologia requer o desenvolvimento de um método de compactação e descompactação em tempo de execução, similarmente ao que EDD possui. Uma possibilidade seria algum compactador padrão aberto baseado em zip, mantendo assim a Open-EDD uma solução aberta e não-proprietária.

Outro problema relacionado é o desempenho requerido para consultar informações dos arquivos de Open-EDD. Para evitar grandes tempos para processamento, um mecanismo foi criado para interpretação da Open-EDD: Quando o sistema necessita acessar dados de um tipo de dispositivo específico (relacionado um arquivo Open-EDD), toda a informação deste (uma estrutura em árvore - DOM) é carregada em objetos estáticos. Então, quando o sistema necessita acessar dados desse tipo de dispositivo novamente, a consulta é feita instantaneamente porque tais informações já estão em memória. O usuário do sistema necessita aguardar apenas alguns poucos segundos (1 a 4 segundos) para carregar a estrutura XML a primeira vez e, posteriormente, todas as operações envolvendo o mesmo tipo de dispositivo são executadas instantaneamente, não necessitando a leitura dos dados do disco novamente.

10 CONCLUSÕES

A tecnologia Open-EDD é mais fácil de ser manipulada que a EDD e FDT/DTM. Tal facilidade é notada pelos desenvolvedores de dispositivos tanto para desenvolvedores de aplicativos IHM. Além disso, é possível de ser integrada de forma fácil em qualquer sistema por ser uma tecnologia aberta. Tal fato pode ser evidenciado se, por exemplo, fosse experimentada a integração da tecnologia FDT/DTM no simlador FBSIMU. Como o simulador é executado no ambiente LabView, seria árdua ou até mesmo impossível a integração de um componente baseado em COM com a complexidade inerente à essa tecnologia. A tecnologia EDD por sua vez, também resultaria no mesmo problema, já que o interpretador DDS é feito em linguagem C e a interface de comunicação é feita através de estruturas de baixo nível.

A proposta de uma tecnologia de descrição de dispositivos aberta e não-proprietária como a Open-EDD pode incorporar todos os recursos da EDD juntamente com a importante característica de eliminar a necessidade de aquisição dos chamados *toolkits* de desenvolvimento.

De acordo com a filosofia de sistemas abertos, este estudo apresenta a Open-EDD como uma alternativa aberta, não como uma substituição da EDD ou a FDT/DTM. Em alguns ambientes, tais como de ferramentas comerciais, talvez o FDT/DTM ou a EDD ou a junção dos dois sejam mais adequados por terem mais recursos implementados (mesmo que sejam proprietárias e soluções fechadas) e por serem apoiadas por fundações internacionais de influência, tais como a Fieldbus Foundation^{TM}, HART Foundation^{TM}, PROFIBUS Organization e OPC Foundation^{TM}.

A utilização de XML para ser a base de uma tecnologia de dispositivos possibilita o mecanismo de geração de interface gráfica automática e padrão para todos os dipositivos de campo. Esse fato facilita o "*look and feel*" do usuário, já que terá o mesmo estilo e padrão para analisar os dados de qualquer dispositivo.

Em razão da extensibilidade provida pela tecnologia XML, a Open-EDD pode estender a descrição para outros protocolos, desde que tenham informações similares como, por exemplo, no caso dos protocolos FF, HART e PROFIBUS.

Este estudo foi validado numa ferramenta comercial e numa ferramenta de uso acadêmico que comprova que a técnica é praticável e pode ser aplicável em qualquer outro ambiente.

AGRADECIMENTOS

Os autores gostariam de agradecer a empresa Smar Equipamentos Industriais e a Escola de Engenharia de São Carlos pelo incentivo e pela estrutura oferecida.

REFERÊNCIAS

Apache Software Foundation (2005). <http://xml.apache.org/xerces-c/>. Acessado em Outubro, 2005.

Booch, G. (1994). *Object-oriented analysis and design with applications*. 2nd.ed. California: Benjamin/Cummings.

Booch, G.; Rumbaugh, J.; Jacobson, I. (1999). *The unified modeling language user guide*, Rading Mass: Addison-Wesley.

Borst, W. (1991). Kommunikation auf der instrumentierungsebene. *Elektronik*, Munchem, v.40, n.18, p.72-80.

Donaires, O.S. (2004). FF DD x FDT/DTM: origens, características e tendências. *Controle & Instrumentação*, São Paulo, v.99, p.40-46.

FDT Group (2006). <www.fdt-jig.org>. Acessado em Maio, 2006.

Fieldbus Foundation (1999a). *FF-900-1.5*: Foundation specification device descriptiong, part 1. Austin.

Fieldbus Foundation (1999b). *FF-103-1.7*: Foundation specification common file formatg, part 1. Austin.

Fieldbus Foundation (1999c). *FF-800-1.4*: Foundation specification system architecture. Austin.

Fieldbus Foundation (1999d). *FF-890-1.3*: Foundation specification function block application processg, part 1. Austin.

Kasemann, J (2004). Profinet basics. <www.interclub.com/Profinet Basics.pdf>. Acessado em Outubro, 2005.

Kastner, W.; Kastner-Masilko, F. (2004). EDDL inside FDT/DTM. *Proceedings of the IEEE International Workshop On Factory Communication Systems*, Vienna, pp.365–368.

Merrit, R. (2002). FDT is not a fieldbus panacea. *Control*. http://www.easydeltav.com/news/viewpoint/fdt.pdf>. Acessado em: Novembro, 2005.

Microsoft – COM (2008). <http://www.microsoft.com/com/default.mspx>. Acessado em Maio, 2008.

Miller P. (2002). Interoperability what is it and why should I want it? <http://www.ariadne.ac.uk/issue24/interoperability>. Acessado em Outubro, 2006.

Monaco, F.J. (2006) Non-proprietary knowledge management. <http://www.icmc.usp.br/∼monaco>. Acessado em Agosto, 2006.

National Instruments (2007). Using the LabVIEW Shared Variable. <ttp://zone.ni.com/devzone/cda/tut/p/id/4679>. Acessado em Novembro, 2007.

Netbeans (2006). <http://www.netbeans.org>. Acessado em Fevereiro, 2006.

OPC Foundation (2006). <http://www.opcfoundation.org/>. Acessado em Fevereiro, 2006.

Pantoni, R. P. (2006). Desenvolvimento e implementação de uma descrição de dispositivos aberta e não-proprietária para equipamentos FOUNDATION fieldbus baseada em XML. Dissertação de Mestrado – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

Pantoni, R. P. ; Passarini, L. C. ; Brandao, D. (2007). Developing and implementing an open and non-proprietary device description for fieldbus devices based on software standards. *Proceedings of the IEEE International Symposium on Industrial Electronics*, 2007, Vigo, pp. 1887-1892.

Pantoni, R. P. ; Brandao, D. ; Torrisi, N. ; Mossin, E. A. (2008). Integration of an Open and Non-proprietary Device Description Technology in a Foundation Fieldbus Simulator. *Proceedings of the 7th IEEE International Workshop on Factory Communication Systems*, Dresden, v. 1, pp. 435-443.

Pinotti Jr, Mario ; Brandão, D. (2005) A flexible fieldbus simulation platform for distributed control systems laboratory courses. *The International Journal Of Engineering Education*, Dublin, v. 21, n. 6, p. 1050-1058.

Profibus International (2005). *Specification for PROFIBUS*, device description and device integration: GSD. Germany. v.1.

Profibus International (2005). *GSDML specification for profinet IO*. Germany. v.1.

Smar Equipamentos Industriais (2005). <http://www.smar.com>. Acessado em Outubro, 2005.

School of Computing and IT (2007). <http://www.scit.wlv.ac.uk/∼jphb/cp2101/week4/XML_Parsing.html>. Acessado in Março, 2007.

Simon, R.; Riedl, M.; Diedrich, C. (2003). Integration of field devices using field device tool (FDT) on the basis of electronic device descriptions (EDD). *Proceedings of the IEEE International Symposium on Industrial Electronics*, Pusan, v.1, pp.189 - 194.

Zielinsky, M. (2005) Digital fieldbus installations use EDDL for simplicity with advanced, full functionality. *Computing & Control Engineering Journal*, London, v.15, n.6, pp.24-31.

Thiruvathukal, G. (2004) XML and computational science. *Computing in Science and Engineering*, v.6, n.1, p.74-80.

Walmsley P. (2002). *Definitive XML Schema*. Prentice Hall PTR, Upper Sandle River, p. 03-14.

Wang, Z.; Yu, H.; Wang, H.; Xu A.; Zhou, Y. (2004) The research of XML-based extensible device description for FF. *Proceedings of the Annual Conference Of IEEE Industrial Electronic Society*, Busan, pp.2600-2603.

Wollschlaeger, M (1999). CANopen device descriptions using general purpose modeling languages. *Proceedings of the International CAN Conference*, Turin., pp.03-06 - 03-13.

Yong, L.; Hai-Bin, Y.;Tian-Ran, W.;Zhi-Jia, Y.; (2004). Fieldbus interoperation technologies. *Proceedings of the International Congress On Intelligent Control And Automation*, Hangzhou P.R.China, pp.3620-3623.

XML DOM (2007). <http://www.w3.org/DOM/>. Acessado em Março, 2007.

XML Schema (2006). <http://www.w3.org/XML/Schema>. Acessado em Maio, 2006.