Uma Abordagem de Gerenciamento Integrado de Processos Industriais Empregando a Arquitetura Orientada a Serviços

R. A. Dias, T.E. de Souza, *Núcleo de Engenharia de Redes e Sistemas Distribuídos (NERsD), Instituto Federal de Santa Catarina*

***Abstract* — The integration among the programmable logic controllers and the supervision and control software's has been a critical operation of great importance for industrial production management. The goal of this work is a model development of integration framework for industrial processes using the best practices of Service Oriented Architecture (SOA) using a recent Devices Profile for Web Services (DPWS) as integration technology. The main result is a prototype system used in a small-scale industrial process of bottles filler. Through the scenario was possible to validate the model feasibility in an industrial environment, doing the performance test to evaluate the model and to observe the characteristics of notification, description, discovery, reporting and control.**

I. Introdução

Com a crescente necessidade de levar informações de maneira rápida e precisa do chão de fábrica para os centros de tomadas de decisões, surgem conceitos inovadores de gerenciamento da produção para aumentar a flexibilidade, modularidade e reconfiguração de sistemas. Tais conceitos tiram proveito das tecnologias de informação e comunicação.

O futuro da indústria de manufatura pode ser caracterizado principalmente pela mudança frequente das demandas do mercado, a pressão do tempo de mercado, o surgimento contínuo de novas tecnologias e, sobretudo, a competição global. Desta forma, é de suma importância a adoção de tecnologias flexíveis e adaptáveis a tais mudanças [1][2].

Atualmente, um terço do custo total de uma fábrica é gasto em instalação e manutenção. Sempre que um processo industrial é alterado devido à mudança de produto produzido ou alteração de suas características, deve ser realizada uma reconfiguração que causa uma inatividade considerável na produção, atuando diretamente nos custos operacionais [1].

As tecnologias existentes contribuíram até o momento para a evolução dos processos industriais, mas suas características não contemplam as novas necessidades que atualmente as indústrias demandam.

O autor R. A. Dias é professor do Departamento de Metal Mecânica do Instituto Federal de Santa Catarina. Atua nos cursos de graduação e mestrado em Mecatrônica. É líder do NERsD.

O autor T.E. de Souza é graduado no curso Superior de Tecnologia em Sistemas Eletrônicos e bolsista do NERsD.

Segundo Souza [2], o principal problema é que essas tecnologias dependem de que cada dispositivo tenha um controlador (*driver*) para se comunicar com um sistema de banco de dados. Esse *driver* fornece informações para a comunicação de um dispositivo com um determinado sistema, tornando essas soluções limitadas aos sistemas suportados pelo *driver* e a dispositivos ou recursos que suportem a conectividade entre o dispositivo e o banco de dados.

Surge então a necessidade de uma tecnologia que propicie a integração das informações de chão de fábrica com os sistemas de informação gerencial de forma direta e transparente, independente de plataforma. Uma alternativa recente é o emprego da Arquitetura Orientada a Serviços (*Service-oriented architecture* – SOA) que, conforme Papazoglou [3], "é uma caracterização de sistemas distribuídos, em que as funcionalidades do sistema são expostas via descrição de uma interface, permitindo a publicação, localização e a invocação por meio de um formato padronizado".

A SOA tem por maior objetivo a capacidade de conectar uma ampla variedade de sistemas sem uso de programas proprietários, a fim de alcançar a interoperabilidade verdadeiramente aberta. SOA por si só é somente um conceito.

Imaginam-se dois programas que foram escritos em diferentes linguagens de programação e em sistemas operacionais diferentes. SOA vem para facilitar a integração destas aplicações pela adoção de uma arquitetura aberta e independente de plataforma.

Dentro deste contexto, este trabalho tem por objetivo o desenvolvimento de um modelo para a integração dos dispositivos existentes no ambiente industrial com os softwares de controle e supervisão, utilizando os conceitos da Arquitetura Orientada a Serviços através da utilização do Perfil de Dispositivos para Serviços *Web*. O modelo será validado em um cenário, onde serão estudadas as principais características da tecnologia e verificado o seu desempenho.

II. Tecnologia DPWS

O DPWS define um conjunto mínimo de funcionalidades que permite a dispositivos com limitações de recursos computacionais suportarem SW. Este conjunto mínimo que é formado basicamente por troca de mensagens seguras, serviços de descoberta e de notificação de eventos e, ainda, pela

descrição de serviços, deve permitir a implementação direta em dispositivos embarcados em geral, sem comprometer a conformidade com a padronização de SW.

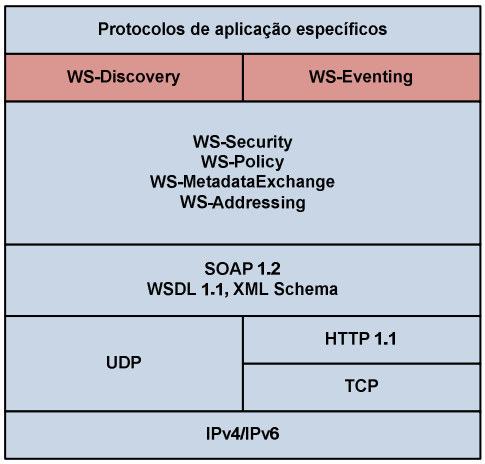

Fig. 1. Pilha de protocolos da arquitetura de DPWS.

A. *Processo de padronização*

Inicialmente produzido por um consórcio entre Microsoft, Ricoh, Intel e Lexmark, atualmente padronizado pela OASIS, que através de um comitê técnico sob o codinome WS-DD, especifica além do DPWS o SOAP-over-UDP e o Web Services Dynamic Discovery (WS-Discovery) [4]. Atualmente as três especificações estão na versão 1.1.

B. *Arquitetura DPWS*

O DPWS possui a pilha de protocolos mostrada na Fig. 1, e é composta dos protocolos padrão dos SWs e algumas extensões. Um detalhe importante, mostrado na Fig. 1, é que a especificação OASIS [4] para o DPWS define que o mesmo trabalhe com o protocolo SOAP na versão 1.2 e a WSDL na versão 1.1.

C. *Modelo DPWS*

O modelo computacional enfatizado na especificação DPWS preconiza que dispositivos podem assumir diferentes papeis: são ou consumidores de serviços (clientes), ou serviços ou, ainda, ambos. No caso de serviços, dois tipos são distinguidos: serviços de hospedagem (*hosting services*) e serviços hospedados (*hosted services*). A figura 2 ilustra como os dois tipos de serviços se enquadram no modelo.

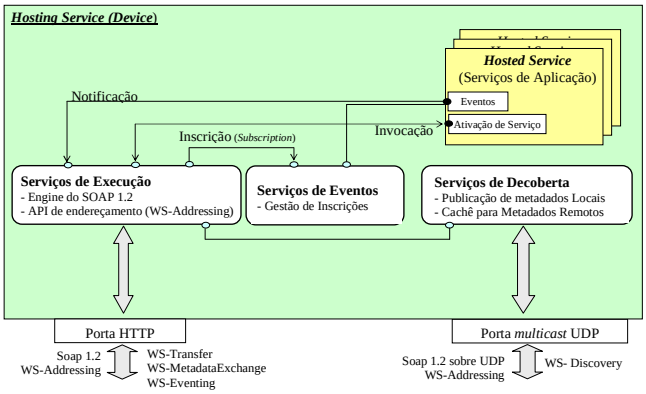

Fig. 2. Modelo computacional do DPWS

O chamado serviço de hospedagem (ou dispositivo) é uma parte importante do modelo DPWS. Vários dos aspectos não funcionais que atuam na evolução de serviços de aplicação estão concentrados neste componente, na forma de serviços incluídos. Serviços que permitem descoberta dinâmica e a troca de metadados (interfaces WSDL e seus *attachments* como o WS-Policy, XML Schemas, etc) são exemplos destes serviços fundamentados no componente dispositivo. O próprio motor do protocolo SOAP 1.2 é também parte destes serviços do componente de hospedagem.

Os serviços hospedados são SW específicos das aplicações e fornecem o comportamento funcional das mesmas. Um dispositivo pode possuir diversos serviços hospedados, cada um deles possuindo seu *endpoint* próprio e é daí que vem o nome serviço de hospedagem. Estes serviços hospedados dependem do dispositivo e fazem uso dos seus serviços incluídos tal qual o processo de descoberta [5].

O repertório destes serviços incluídos envolve o conjunto de quatro áreas: descoberta, eventos, descrição e trocas de mensagens.

Os Serviços de Descoberta Dinâmica (WS-Discovery [6]) são usados por dispositivos para se anunciarem em uma rede e para serem descobertos por clientes. O WS-Discovery usa o SOAP sobre a pilha UDP/Multicast IP para difundir e escutar mensagens de descoberta. Em WS-Eventing [7], são definidos serviços de Publicação (fonte de eventos) e Inscrição em uma fonte de eventos. A combinação de serviços de aplicação com estes serviços incluídos permitem clientes se inscreverem para recepção de mensagens assíncronas (eventos) produzidas por serviços hospedados. Os metadados de serviços hospedados ficam disponíveis para clientes através do uso das especificações do WS-MetadataExchange [8]. As trocas de mensagens ocorrem seguindo as especificações SOAP 1.2, como informado antes. As informações de cabeçalho do SOAP seguem o WS-Addressing [9], permitindo a disponibilidade destas para qualquer protocolo de transporte (HTTP, SMTP, TCP, UDP, etc.).

D. Plataformas de desenvolvimento

Existem diversas ferramentas que auxiliam no desenvolvimento de aplicações baseadas em DPWS. Eles surgiram antes mesmo do início do processo de padronização assumido pela OASIS.

O sistema operacional Windows Vista provê suporte nativo ao DPWS através da WSDAPI (*Web Services on Devices API*). Como resultados do projeto ITEA SIRENA, outras três ferramentas foram criadas e estão disponíveis sob licenças de código aberto. O *Web Services for Devices* (WS4D) possui três implementações de pilha DPWS, uma em linguagem C e outras duas em *Java*, sendo uma delas para *Apache Axis*.

III. GERENCIAMENTO DE PROCESSOS INDUSTRIAIS EMPREGANDO SOA

Um dos problemas encontrados na indústria é a integração dos controladores lógicos programáveis (CLPs) com ferramentas SCADA (*Supervisory Control and Data Acquisition*) e outros CLPs. Essa integração é feita através de drivers que precisam ser desenvolvidos pelos desenvolvedores de cada sistema SCADA.

Uma das soluções existentes para integração de redes industriais é o uso de *Object linking and Embedding for Process Control* (OLE *for process control* ou simplesmente OPC). O padrão OPC fornece um meio comum para conectar fontes de dados como CLPs, dispositivos de automação e banco de dados com um servidor de aplicações. O padrão OPC conecta as fontes de dados (dispositivos de automação, banco de dados) com as aplicações. Neste contexto, atua como elemento de ligação entre aplicações SCADA e os CLPs, por exemplo. O OPC é baseado na tecnologia *Distributed Common Object Model* (DCOM) (Modelo da *Microsoft* para computação distribuída), permitindo a integração de aplicações industriais e o compartilhamento de seus dados em forma de objetos localmente através de uma *Local Area Network* (LAN) [10].

Entretanto, muitas características inerentes ao DCOM prejudicam o desempenho das aplicações em algumas situações. Algumas de suas desvantagens são:

- é dependente de plataforma [10][11][12];
- o tipo de mensagens geradas é muito complexo [10];
- ao mandar mensagens pela internet, torna-se um grande problema a presença de firewalls [10][11][12];

Outra tecnologia parecida com o DCOM é o *Common Object Request Broker Architecture* (CORBA) que é independente de plataforma, mas contêm algumas deficiências do DCOM entre outras [10].

Dentro desse contexto, a integração dos CLPs com os demais dispositivos do ambiente industrial, utilizando tecnologia independente de plataforma, com recursos de: auto-descrição de suas funcionalidades, auto-descoberta, e com orientação a eventos, é a principal questão deste trabalho.

A. Por que SOA?

A SOA implementada através de serviços *Web* tem se tornado um padrão de desenvolvimento para a maior parte das empresas. Investimentos de corporações como a *Microsoft* e a *Sun* (*Oracle*), criaram ferramentas de desenvolvimento poderosas que facilitaram o desenvolvimento de serviços *Web*, que deixou de ser uma aposta, para se tornar um negócio que tem previsão de gerar 30 bilhões de dólares e estar presente em 80% das aplicações das grandes corporações até o final do ano de 2010. Estimativas apontam serviços *Web*, como a principal tecnologia para integração de sistemas distribuídos por corporações de tecnologia de informações [13]

Essa migração de tecnologias, como DCOM, CORBA ou outros *middlewares*, para serviços *Web* [12], deve-se a algumas vantagens que estão intimamente ligadas aos principais conceitos adotados por serviços *Web*.

A primeira vantagem é o baixo acoplamento entre o cliente e o servidor. Ao utilizar SOA, o serviço *Web* é desenvolvido para que possa ser consumido pelo cliente com pouco ou nenhum conhecimento do contexto existente no serviço *Web*, possibilitando a interação entre cliente e servidor, utilizando apenas as operações existentes no serviço Web. Quanto mais baixo for o acoplamento mais fácil será a reutilização de código [13].

Outra vantagem é que serviços *Web* utilizam protocolos e padrões que estão realmente consolidados. O XML é usado como modo de formatação da informação e utiliza o protocolo HTTP como transporte para essas informações. Como o HTTP é suportado por qualquer plataforma que permita a visualização de páginas na internet e, o XML é um padrão de transporte e armazenamento de dados que utiliza texto plano, os serviços Web se tornam independentes de plataformas ou sistemas [10].

Como citado anteriormente, a utilização dos preceitos da SOA através de serviços *Web*, como ferramenta de integração entre sistemas distribuídos, já é largamente utilizada nos departamentos de desenvolvimento de tecnologia de informações (TI). Mas a idéia de utilização de serviços para a comunicação de sistemas embarcados está apenas começando.

Através da SOA, este trabalho tem a finalidade de expandir a integração entre os softwares SCADA e SAD (Sistemas de Apoio à Decisão) em nível de chão de fábrica, disponibilizando serviços embarcados nos CLPs.

Ao tratar o CLP como um serviço *Web* embarcado, ele terá as suas funcionalidades expostas, eliminando-se a necessidade de drivers de dispositivo, ou tecnologias de integração como o OPC. Procedimentos de configuração ou reconfiguração poderão ser automatizados, em grande parte, através das funcionalidades de autodescoberta e autodescrição inerentes aos serviços.

Como exemplo, uma fábrica de bebidas usa uma das suas esteiras para encher garrafas de 600 mililitros, mas devido à demanda das garrafas de 355 mililitros, é resolvido utilizar a esteira para encher garrafas de 355 mililitros. Para isto, dever-se-á configurar o tempo que cada garrafa será cheia. Esta

configuração pode ser feita diretamente pelo software de controle e supervisão, através de uma operação existente no serviço Web embarcado no CLP.

Outro exemplo é o caso da substituição de um CLP devido a um problema qualquer. Através do serviço de descoberta, as características do novo CLP são obtidas. Assim, é possível realizar-se a troca, com a seleção automática do Serviço Web de configuração correspondente ao novo CLP, sem a necessidade de drivers ou do OPC. Quando o novo CLP for instalado e ligado a rede, será enviada uma mensagem para o software SCADA anunciando o novo CLP e descrevendo suas funcionalidades, que poderão ser agregadas pelo programa de controle e supervisão.

IV. SOLUÇÃO PROPOSTA

A. *Gerência Integrada de Processos Industriais utilizando SOA (GIPInSOA)*

O modelo apresentado na Fig. 3 é baseado nos conceitos de SOA no âmbito de anunciar, descobrir e descrever dinamicamente os serviços existentes em cada elemento do ambiente industrial. Para isso, é amplamente utilizado o DPWS como tecnologia de integração dos diversos elementos.

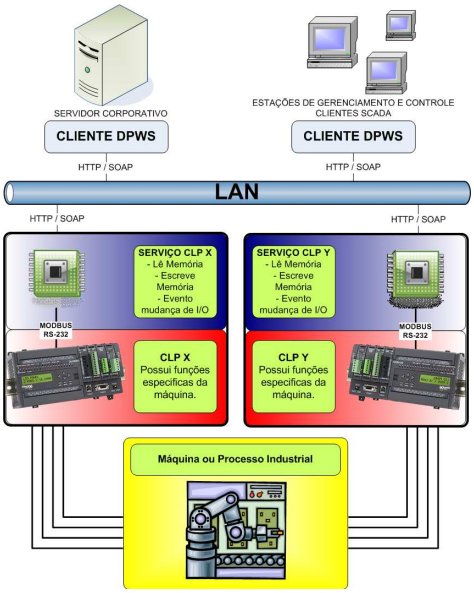

Fig. 3. Modelo computacional do DPWS

Ao embarcar um serviço Web em um dispositivo da indústria, como um CLP, se cria o conceito de um driver "universal", que suporta qualquer sistema operacional ou plataforma de software. Assim que o CLP for ligado à rede IP, ele será anunciado, e os programas interessados terão sua descrição para poderem utilizá-lo.

B. *Plataforma de Desenvolvimento*

Como plataforma de desenvolvimento para o serviço web embarcado foi utilizado o kit *Meridian/p* da *Device Solutions* que possui um micro-controlador ARM9 e o .NET *Micro Framework* (NETMF).

O NETMF foi desenvolvido pela *Microsoft* como uma plataforma para desenvolvimento de sistemas embarcados, com recurso de no mínimo 250 KB de RAM. O principal objetivo é o desenvolvimento de sistemas embarcados para linguagens de alto nível, tornando o desenvolvimento mais rápido de aplicações embarcadas e agilidade no trabalho de manutenção de código.

O desenvolvimento das aplicações utilizando NETMF pode ser realizado através do programa *Visual Studio*, ferramenta amplamente difundida no desenvolvimento de aplicações para Windows e Internet. Através do *Visual Studio*, por exemplo, são fornecidas ferramentas de gravação, de *debug* e de emulação.

Ao lançar o NETMF 4, em novembro de 2009, a *Microsoft* liberou o seu código através da licença *apache* 2.0, que permite o uso e distribuição do código fonte e da aplicação final, livre de royalties. O que trará contribuições para a tecnologia de fora da *Microsoft*, o que ajudará no desenvolvimento do *framework* [14].

C. *Cenário de Validação*

Como cenário de validação para o modelo GIPInSOA da Fig. 3, foi concebido um processo industrial para envasamento de garrafas.

A Fig. 4 mostra em detalhes a estrutura do cenário de validação. O processo industrial é simples, contém dois sensores, um temporizador no CLP e um motor. Apesar da sua simplicidade, é suficiente para estudar as características do modelo.

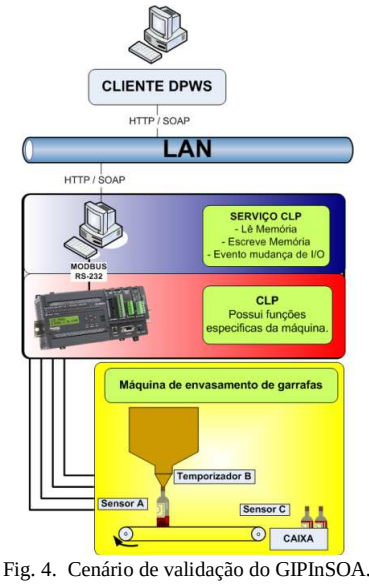

Fig. 4. Cenário de validação do GIPInSOA.

Basicamente o cenário consiste em uma sequência de procedimentos. A esteira gira até o ponto que a garrafa seja detectada pelo Sensor A. Depois, é iniciado o enchimento da garrafa que é concluído com o estouro do Temporizador B. A esteira gira novamente até o Sensor C que indica que a garrafa

foi para a caixa, concluindo o processo.

D. *Serviço Web Embarcado*

O serviço Web foi desenvolvido através das bibliotecas fornecidas pelo .NET *Micro Framework* (NETMF) da Microsoft sobre o hardware *Meridian/P* da *Device Solutions*.

Foi modelado de forma que fosse possível adquirir e enviar informações para o CLP. Para evitar varreduras entre o cliente DPWS e o CLP foi criado um evento para leitura de memória que é disparado quando determinada posição de memória tem o seu valor alterado. Na Fig. 5, podem ser observados os métodos utilizados.

Fig. 5. Classe representando o serviço Web do CLP.

O serviço foi modelado na forma da classe “clpService”, que herda as características de um serviço hospedado (“DpwsHostedService”).

E. *Cliente DPWS*

Para a implementação do cliente DPWS no computador com o sistema operacional *Windows* XP foram enfrentados alguns problemas. A *Microsoft* disponibiliza a (*Web Services on Devices* - API) WSDAPI somente a partir do *Windows Vista*. Foram despendidas várias horas com tentativas sem sucesso de tornar a API funcional no *Windows* XP.

SOA4D foi outra ferramenta para implementação do cliente que foi estudada, mas no final foi decidido pela implementação do núcleo funcional do cliente totalmente em C#. Os motivos foram, além da facilidade da programação em alto nível disponível pela linguagem, foi a oportunidade de estudar profundamente a troca de mensagens e funcionamento do DPWS.

Para desenvolver um cliente DPWS, foi basicamente necessário ter um cliente HTTP para o envio de mensagens, um servidor HTTP para recebimento de mensagens e um tratamento para mensagens XML. Tudo organizado de forma que seja possível a interpretação das mensagens SOAP de controle e de eventos.

V. RESULTADOS

Neste capítulo serão apresentados os resultados do modelo proposto através da implementação do cenário de validação. O cenário é composto por uma aplicação cliente em um computador pessoal, um serviço *Web* embarcado em um dispositivo, um CLP que suporta o protocolo MODBUS e um aparato, representando uma máquina envasadora de garrafas, contendo sensores e luzes.

A. *Protótipo*

A Fig. 5 mostra o funcionamento da envasadora de garrafas. O funcionamento do processo foi programado no CLP em linguagem *Ladder*. O serviço DPWS, como proposto inicialmente, servirá apenas para o gerenciamento e supervisão do processo.

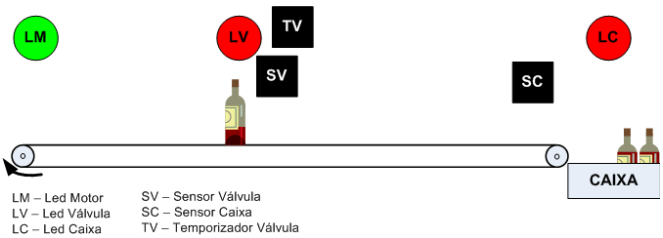

Fig. 5. Esquema da envasadora de garrafas.

O processo começa com o motor ligado sinalizado pelo led do motor (LM), que gira até a garrafa atingir o sensor da válvula (SV). Com o motor parado, é aberta a válvula (LV) durante um tempo pré-programado no temporizador de válvula (TV). O tempo em TV é o que define a quantidade de líquido a ser carregado na garrafa. Quando o temporizador alcançar o valor esperado, a válvula será fechada e o motor continuará girando até o sensor da caixa (SC), o que indica que a garrafa está dentro da caixa. Por último quando a última garrafa for colocada na caixa será ativado o led da caixa (LC), que indica que caixa está cheia.

O serviço DPWS é responsável pelo controle e supervisão e suportará as seguintes funcionalidades:

- regulagem do tempo de válvula aberta; quantidade de garrafas por caixa;
- início da produção;
- parada da produção;
- envio de notificações a cada caixa cheia concluída.

Para gerar a notificação de “caixa cheia concluída” foi realizada uma varredura da posição de memória que conta as garrafas já envasadas entre o serviço DPWS e o CLP.

O cliente DPWS foi o responsável por realizar a descoberta do serviço DPWS, enviar as mensagens de controle e receber as notificações de caixa concluída.

Sempre que um dispositivo que contém um serviço DPWS é ligado na rede local, ele envia uma mensagem *hello* na forma de *multicast* para qualquer cliente DPWS presente na rede. A

parte do programa cliente DPWS responsável por receber esta mensagem de anúncio no programa cliente DPWS foi denominado de "*Explorer*". Nele são listados os serviços DPWS da rede local e obtidas informações detalhadas a respeito de cada um deles.

No "*Explorer*" do cliente é possível realizar uma sondagem da rede (*probe)* para verificar a existência de todos os dispositivos presentes na rede local, uma mensagem de resposta do tipo *resolve* verifica a existência de determinado dispositivo e uma mensagem do tipo *get* obtém a descrição de dispositivo, modelo, serviço de hospedagem e serviços hospedados. No "Explorer" a mensagem *hello* do serviço DPWS é automaticamente reconhecida como um dispositivo.

Depois do "Explorer" do cliente DPWS descobrir ou capturar o anúncio de um serviço CLP o cliente DPWS poderá consumir o serviço através da processo denominado "Envasador", onde ocorrerá o controle e supervisão do processo industrial. Aqui são utilizadas as principais operações disponíveis pelo serviço DPWS. Através dele é possível obter e definir o tempo de abertura de válvula e quantidade de garrafas que serão armazenadas por caixa. Também é possível obter a quantidade de garrafas por operação de controle. A quantidade de caixa é alimentada através de um evento disparado a cada caixa concluída.

A Fig. 6 mostra a tela do controle e supervisão no nível de gerência da envasadora de garrafas.

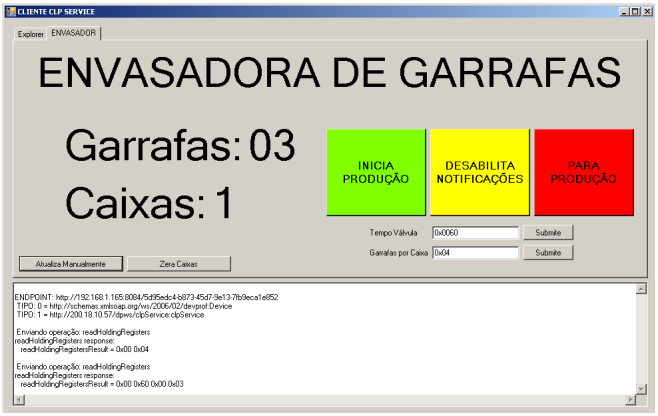

Fig. 6. Controle e supervisão da envasadora de garrafas.

O conjunto hardware e CLP é responsável pela parte funcional da envasadora de garrafas. O CLP executa um programa que recebe sinais dos sensores e atua no motor e na válvula do hardware, como explicado anteriormente neste mesmo capítulo na Fig. 5.

A Fig. 7 é uma foto do conjunto CLP e hardware, onde são vistos os *bournes* de alimentação, os sensores e os *leds* indicando se o motor ou a válvula estão ativos.

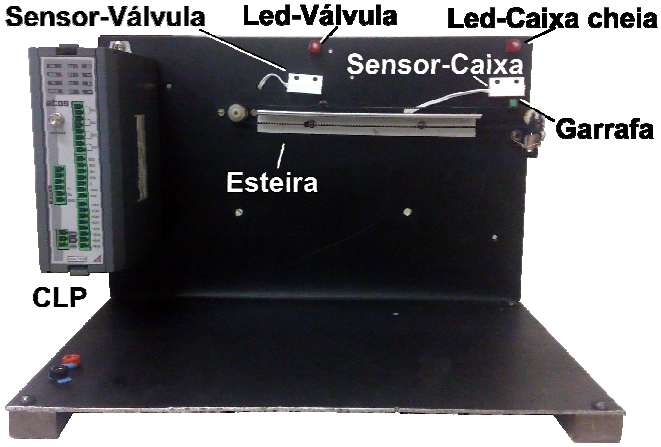

Fig. 7. Aparato para emular o processo de envasamento de garrafas.

B. *Avaliação de desempenho*

Para realizar a avaliação de desempenho das trocas de mensagens do DPWS, foi utilizado como ferramenta para medição o software Wireshark. Este software é um analisador de rede, mais conhecido como *sniffer*, que captura os pacotes enviados e recebidos da placa de rede, documentando o momento de chegada ou envio de determinado pacote.

Com esta ferramenta foi cronometrado o tempo entre o envio e a resposta das mensagens do DPWS. Os resultados podem ser observados na Tabela I.

Tabela I – Tempo entre troca de mensagens.

Operação	Tempo Conexão (segundos)	Tempo requisição / resposta (segundos)	Tempo Total (segundos)
Probe	**Sem conexão**	**1.180**	**1.180**
restart (oneWay)	**0.355**	**0.547**	**0.902**
readHolding-Registers (twoWay)	**0.357**	**0.872**	**1.220**
presetSingle-Register (twoWay)	**0.349**	**0.798**	**1.147**
subscribe	**0.361**	**1.109**	**1.471**
Register-Changed (Eventing)	**0.023**	**0.052**	**0.211**

Observando a Tabela I, pode-se verificar que se tratando de mensagens de controle o DPWS é lento, levando mais de um segundo para mensagens de controle simples. Mas as mensagens de eventos, por evitar as varreduras na rede, são muito rápidas levando apenas 212 milissegundos para serem completadas.

Foram realizadas medições com o mesmo aparato, mas utilizando-se um servidor OPC no lugar de Serviços Web e constatou-se que a leitura da *tag* (variável) consome em torno de 480 milissegundos, valores similares foram encontrados na literatura [15]. Este tempo confrontado com o readHolding-Registers (twoWay) é mais rápido. No entanto o OPC não dispõe de serviço de notificação de eventos como no DPWS.

A utilização de notificação de eventos para supervisão do processo utiliza protocolo UDP, o que justifica a baixa latência verificada na tabela I. Assim, com este recurso o uso do DPWS se torna bastante viável.

VI. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Neste trabalho foi realizado um estudo sobre redes industriais, protocolos de comunicação e tecnologias de sistemas distribuídos. Através disso, foi desenvolvido um modelo de integração entre os dispositivos industriais e os softwares de controle e supervisão.

O modelo foi concebido utilizando os preceitos de SOA através da utilização do DPWS. O protótipo consiste que o dispositivo a ser integrado com um sistema SCADA se torne um serviço. Com o serviço, o dispositivo passa a ter suas características expostas, podendo ser consumidas sem a necessidade de drivers proprietários ou outras tecnologias de integração.

O uso do DPWS provê diversas características interessantes para a integração de dispositivos, dentre elas estão o anúncio, descoberta, descrição, notificação e controle.

Depois de elaborado, o modelo foi validado através de um processo industrial em pequena escala para envasamento de garrafas. Através deste cenário foi possível verificar o funcionamento do modelo, testar as características de anúncio, descoberta e descrição, e realizar os testes de desempenho da tecnologia.

O desenvolvimento da estrutura básica do serviço DPWS foi realizado através da WSDL utilizando as ferramentas de geração de código do NETMF, dispensando a maior parte do trabalho complexo de programação. Depois da estrutura pronta, foram desenvolvidas as operações que o serviço deveria disponibilizar.

Como trabalhos futuros sugere-se:

- O desenvolvimento de conversor ou gateway de comunicação entre as aplicações legadas que utilizam OPC e o DPWS.
- Desenvolvimento de um CLP totalmente baseado em SOA, eliminando a necessidade de um adaptador DPWS / Modbus.

REFERÊNCIAS

[1] F. Jammes, H. Smit, Service-Oriented Paradigms in Industrial Automation. Parallel and Distributed Computing and Networks. Innsbruck, Áustria: 23rd IASTED International Multi-Conference. 2005. p. 8.

[2] L. M. S. Souza, et al. SOCRADES: A web service based shop floor integration infrastructure. Proc. of the Internet of Things (IOT 2008), Springer, 2008. http://people.inf.ethz.ch/mkoehler/papers/IoT08.pdf.

[3] M. P. PAPAZOGLOU: Service-Oriented Computing: Concepts, Characteristics and Directions. Fourth International Conference on Web Information Systems Engineering (WISE'03), Roma, 10 Dezembro 2003. 10. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.77.6394&rep=rep1&type=pdf.

[4] OASIS Web Services Discovery and Web Services Devices Profile (WS-DD) TC, on-line: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-dd, acessado em 17/10/2009.

[5] F. Jammes, A. Mensch, H. Smit, "Service-Oriented Device Communications Using the Devices Profile for Web Services", no 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), Niagara Falls, Ontario, Canada, May 21-May 23. 2007.

[6] Web Services Dynamic Discovery (WS-Discovery), on-line: http://schemas.xmlsoap.org/ws/2005/04/discovery/, acessado em 19/10/2009.

[7] W3C Working Draft, "Web Services Eventing (WS-Eventing)", online: http://www.w3.org/TR/ws-eventing/, acessado em 20/10/2009.

[8] W3C Working Draft, "Web Services Metadata Exchange (WS-MetadataExchange)", on-line: http://www.w3.org/TR/ws-metadata-exchange/, acessado em 20/10/2009.

[9] W3C Recommendation, "Web Services Addressing 1.0 – Core", Maio de 2006, on-line: http://www.w3.org/TR/ws-addr-core/, acessado em: 19/10/2009.

[10] V. KAPSALIS. et al: Architecture for Web-based services integration. The 29th Annual Conference of the IEEE Industrial Electronics Society (IEEE-IECON'03), Virginia, November 2003. 866-871. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.9589&rep=rep1&type=pdf.

[11] FILHO, F. S. D. L. Gerência de Informação de Processos Industriais Usando Web Service. CEFET-RN. Natal (RN), p. 53. 2005. http://www.cefetrn.br/~ajdsouza/tcc/Monografia_TCC_CLP2WEB.pdf.

[12] HENNING, M. The Rise and Fall of Corba. ACM QUEUE, New York, v. 4, n. 5, p. 28-34, June 2006. ISSN 1542-7730.

[13] GANDOLPHO, C. Cresce uso de SOA. Info Corporate, 2008. Disponivel em: <http://info.abril.com.br/corporate/infraestrutura/cresce-uso-de-soa.shtml>. Acesso em: 27 Maio 2010.

[14] GALLI, P. Microsoft to Open Source the.NET Micro Framework. Port 25: The Open Source Community at Microsoft, 16 Novembro 2009. Disponivel em: <http://port25.technet.com/archive/2009/11/16/microsoft-to-open-source-the-net-micro-framework.aspx>. Acesso em: 29 Maio 2010.

[15] MAHNKE, W., LEITNER, S-H., DAMM, M. "OPC Unified Architecture". Springer. 2009. Germany.