COMO DIMINUIR O CUSTO DE INTEGRAÇÃO DA CAMADA MES UTILIZANDO PADRÕES ABERTO E SOA[1]

Eduardo do Carmo Silva[2]
Marcos Alves Filho[3]
Vinicius Matos Paiva[4]

Resumo
Este trabalho mostra uma nova abordagem para realizar a integração entre a camada MES e os demais níveis. O uso de padrões abertos como Web Services, Java e XML, permite que as informações sejam trocadas de forma estruturada e auto-documentada. A orientação a serviços alinhados com o processo da empresa possibilita que as especificidades de cada área sejam tratadas pela automação enquanto os serviços de alto nível são reaproveitados. A partir do levantamento dos processos, foi iniciado o desenvolvimento de um MES, baseado em SOA, em que todos os cadastros corporativos, como as matérias-primas, fluxos de produção, entre outros, estarão em uma base comum e serão consumidos pelo MES, sem replicação de dados. Além disso, a troca de informação entre os níveis, como automação, supervisão, PIMS e o próprio MES será feita utilizando protocolos abertos e SOA. Esta abordagem permitirá que o MES seja, posteriormente, utilizado em outras unidades. Dentre os resultados esperados pela utilização desta nova abordagem orientada a serviços e processos, destacam-se: troca estruturada de dados entre MES, supervisório, PIMS e cadastros corporativos; interface auto-documentada, utilizando padrão aberto, entre MES, supervisório, PIMS e cadastros corporativos; foco no processo de produção das fábricas, sendo o MES um prestador de serviços; reusabilidade da solução com poucas customizações entre as fábricas; centralização de cadastros, sem replicação; customização das fábricas com suas especificidades, mantendo as chamadas a uma interface comum de alto nível e diminuição do custo de desenvolvimento entre as interfaces do sistema MES com os outros sistemas.
Palavras-chave: SOA; Webservice; Integração; MES

AS HAVE LOWER COST OF INTEGRATION OF THE LAYER MES BEING USED OPEN PATTERNS AND SOA

Abstract
This work shows a new approach to do the integration between the MES layer and the other levels. The use of open patterns like Web Services, Java and XML, allow the information to be changed in a structured and self-documented way. The Service Oriented Architecture lined with the company process allows the specificities of each area to be dealt by the automation while the high level services are reutilized. From the survey of the processes, the development of a MES, based on SOA, was initiated. All the corporative data, such as raw materials, production flow, among others, will be in a common base and will be consumed by MES, without data replication. Besides, the exchange of information among the levels, as automation, supervision, PIMS and MES itself will be made using open protocols and SOA. This approach will allow MES to be used later in other units. Among expected results by using this new approach, it is possible to highlight the structured data exchange among MES, supervisory, PIMS and corporative cadastres; self-documented interface, using open pattern, among MES, supervisory, PIMS and corporative cadastres; focus on the production process of the factories, being MES a service provider; reusability of the solution with few customizations among the factories; centering of cadastres, no data replication; customization of the factories with their specificities keeping the calls to a common interface of high level and reduction of the development cost between the interfaces of the MES system with other systems
Key words: SOA; Webservice; Integration, MES

---

[1] *Trabalho técnico apresentado ao X Seminário de Automação de Processos, 4 a 6 de outubro de 2006, Belo Horizonte – MG.*

[2] *Analista de Sistemas da MAGNESITA – eduardos@magnesita.com.br – tel.: (31) 3368.1209.*
[3] *Engenheiro Eletricista da ATAN SISTEMAS – marcos.alves@atan.com.br – tel.: (31) 3289.7733.*
4 Engenheiro de Controle e Automação da ATAN SISTEMAS – vinicius.paiva@atan.com.br – tel.: (31) 3289.7736.

CONSIDERAÇÕES INICIAIS

Com o advento da arquitetura cliente-servidor e a explosão de sistemas departamentais, a diversidade tecnológica espalhada pelas empresas assumiu proporções epidêmicas. Esse emaranhado de sistemas incompatíveis entre si impede que as empresas sejam eficientes e ágeis, atrapalhando os processos internos e a interação com outras empresas.

No entanto, a visão de processos verticalizados, limitados aos departamentos, está sendo substituída por uma visão horizontalizada e integrada. Estes processos extrapolam seus limites físicos e criam “empresas virtuais”, com atividades integradas que são efetuadas por diversas outras empresas parceiras. Interagir e integrar processos entre empresas demanda interoperabilidade entre as aplicações.

Mas como hoje esta interoperabilidade tem sido obtida? Por meio de tecnologias e interfaces proprietárias, com custos altíssimos e com uma demora muitas vezes intoleráveis para as necessidades empresariais.

Entretanto, interoperar sistemas nunca foi fácil. Como as tecnologias são diferentes, sem um padrão aberto, as interfaces entre as diferentes aplicações tinham que ser escritas uma a uma, ou tinha que ser adquirido, para isso, um software específico. O desafio da interoperabilidade gerou uma indústria especializada de consultoria e softwares como EDI (Electronic Data Interchange) e EAI (Enterprise Application Integration).

Ainda assim, estes softwares dependem de interfaces proprietárias. Além disso, a cada novo aplicativo a ser integrado, uma nova interface deve ser instalada ou escrita. Isto implica em mais gastos (novas licenças) e/ou em prazo adicional para escrever a nova interface, pois nem sempre existem interfaces prontas no mercado para toda e qualquer nova demanda de interoperabilidade.

Por que, então, não adotar padrões abertos na interoperabilidade entre aplicações? A idéia é óbvia, mas implementar não é uma tarefa fácil. Existem barreiras tecnológicas e comerciais. Muitos vendedores desenvolvem suas tecnologias e tentam impor esta tecnologia proprietária como um padrão de fato ao mercado, forçando sua adoção pelos outros atores da indústria.

Retornando à busca pela interoperabilidade, alguns passos importantes foram dados nas últimas décadas, que contribuíram, em muito, para chegarmos ao SOA. Um deles foi o advento do conceito de orientação a objetos. Infelizmente, embora o conceito fosse muito interessante, não se conseguiu, na prática, obter a desejada interoperabilidade entre os objetos de diferentes tecnologias. Uma biblioteca de objetos escritos em VisualBasic não interage com uma biblioteca de objetos escritos em Java e vice-versa. E isto, apesar dos esforços de organizações voltadas a especificar padrões, como foi o caso do OMG (Object Management Group), que desenvolveu a especificação CORBA (Common Object Request Broker). Teoricamente, a CORBA foi uma grande idéia, mas, devido à concorrência comercial, acabou não se consolidando. A Microsoft, por exemplo, criou seu próprio “CORBA”, denominado de DCOM (Distributed Component Object Model), que facilita o processo de uso de objetos dentro do mundo Microsoft. Entretanto, não é possível usar este padrão fora das plataformas Microsoft.

Apesar de tudo, as idéias da orientação a objeto ficaram. Por que não identificar as causas de não ter tido sucesso? Obviamente, a falta de padrões abertos foi o principal motivo. Com o advento do padrão XML, que hoje já é uma convenção global aceita por qualquer empresa ou usuário da informática, podemos pensar em uma nova solução para o desafio da interoperabilidade.

Assim, surgiu o conceito de softwares denominados Web Services, que utiliza os padrões abertos da Internet como meio de comunicação e o padrão XML como formato para a troca de mensagens.

O QUE É SOA?

SOA (Services Oriented Architecture) é uma evolução significativa no desafio de resolver um dos principais problemas vividos pelas empresas na área da tecnologia: a habilidade de conectar e integrar sistemas sem depender de softwares e interfaces proprietários. Sua proposta é simples: conectar sistemas por meio de interfaces abertas baseadas no padrão XML (Extensible Markup Language).

Na verdade, a idéia é muito simples: melhor do que criar sistemas no modelo tradicional de aplicação *end-to-end*, a SOA permite o desenvolvimento de um conjunto de "serviços". Estes serviços devem ter inputs e outputs bem definidos e desempenhar funções claras. Com esta abordagem, uma aplicação *end-to-end* pode ser composta por serviços vinculados uns aos outros, que podem ser facilmente utilizados por mais de uma aplicação.

O QUE SÃO WEB SERVICES?

Web Services são softwares aderentes a padrões abertos, que permitem que sistemas em diferentes plataformas de hardware, sistema operacional, linguagens e infra-estrutura de rede comuniquem-se. Isto é possível com o uso, pelos Web Services, dos protocolos Internet (como TCP/IP), que trocam mensagens através de uma interface chamada SOAP (Simple Object Access Protocol), baseada em XML.

É interessante deixar claro que SOA não é o mesmo que Web Service. SOA, como descrito nos tópicos anteriores, é uma arquitetura de troca de informações através de serviços. Já os Web Services podem ser vistos como uma tecnologia capaz de implementar SOA, ou seja, uma ferramenta que materializa o serviço. No entanto, SOA pode ser implementado através de diversas outras ferramentas, como DCOM, serviços nativos do Windows, serviços nativos do Linux, entre outros. A vantagem dos Web Services está na sua especificação e implementação abertas.

COMO IMPLEMENTAR SOA?

Quando uma empresa se compromete com SOA, ela reconhece que terá sempre uma arquitetura heterogênea, formada por múltiplas aplicações de diversos fornecedores que, aparentemente, devem se integrar. Este é o caso da maioria das organizações, que, estrategicamente, optam por não investir todos os seus recursos em uma única tecnologia, até porque a maioria dos fornecedores não possui aplicações completas disponíveis e capazes de satisfazer todas as exigências do usuário final.

Para ser bem sucedido, deve-se identificar qual funcionalidade da aplicação pode ser apresentada como serviço. Uma forma de fazer isso é inventariando as funções de seus aplicativos. Quando se identifica a mesma funcionalidade em múltiplas aplicações, deve-se fazer dela um serviço. Esta abordagem – que começa no nível da funcionalidade – pode dar certo, mas não é o recomendado.

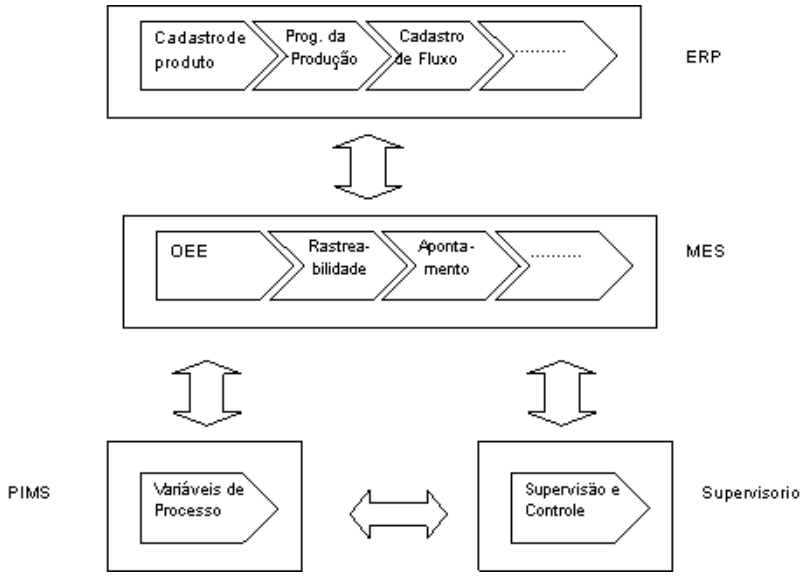
A melhor maneira de identificar candidatos a serviços para um modelo de SOA é por meio do Gerenciamento dos Processos de Negócio (Business Process Management, ou BPM). Com o BPM, o foco começa com seus processos de negócio que, uma vez detectados, permitirão que você identifique claramente qual trabalho precisa ser realizado, por quem ou por qual sistema. A vantagem desta abordagem de desenvolvimento em SOA é que você ganha uma clara compreensão das funcionalidades que estão sendo usadas e com que finalidade, e não apenas o conhecimento de sua existência. No modelo de inventário, você pode ter certeza de que encontrará sobreposições, mas pode também descobrir que realmente está usando uma aplicação que não adiciona valor em outros lugares. Compreender seus processos e determinar as partes da aplicação que são usadas, permitirá que os serviços sejam implementados em um tempo mais maleável, possibilitando compartilhar informações e utilizar completamente toda a arquitetura disponível. Isso, por sua vez, trará melhorias à produção, ampliará a colaboração e reduzirá custos – que são requisitos básicos nas corporações.

HISTÓRICO DO PROJETO

A principal motivação do projeto foi a necessidade de substituição do sistema de automação. Com a substituição, buscava-se diminuir o custo de manutenção, obter uma maior integração com os sistemas de TI da empresa e implementar novas funcionalidades que não eram atendidas pelo sistema existente.

O projeto foi iniciado com uma fase de especificação, em que os processos foram descritos. Verificou-se a existência de processos repetidos, como, por exemplo, a programação de produção, que era feita pelo PCP (Programação e Controle da Produção) e tinha que ser digitada novamente no surpervisório. Todos os requisitos foram levantados e validados com os usuários através de uma especificação. Nesta documentação, foram propostas melhorias no processo e nas soluções de TI e TA utilizadas, atualmente, pela fábrica.

Na nova arquitetura proposta, os vários serviços necessários para a execução do processo são disponibilizados por diferentes sistemas. Enquanto algumas funcionalidades são implementadas no sistema corporativo (ex.: cadastro dos produtos e ingredientes), umas são realizadas no sistema MES (ex.: sugestão de seqüência de abastecimento de silos, rastreabilidade, OEE, apontamento de produção) e outras no PIMS, como o histórico das variáveis de processo e outras dos nível de automação, como a execução da produção e o controle dos equipamentos.



Fonte: Magnesita S. A.

Figura 1. Funcionalidades distribuídas pelos sistemas

Nesta arquitetura, a integração entre os sistemas é feita através dos serviços disponibilizados (Web Services). A popularização dos Web Services foi um fator importante para a viabilização da arquitetura proposta, já que, atualmente, é bastante simples fazer a chamada ao Web Service diretamente a partir do supervisório.

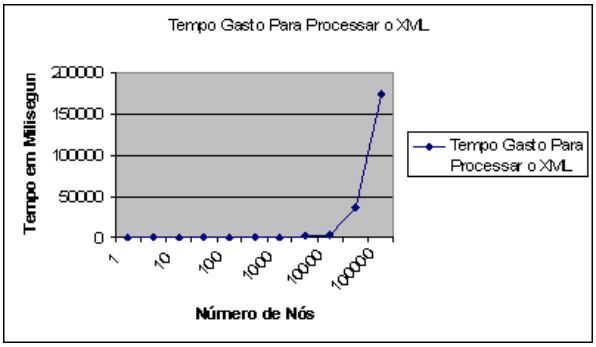
Com a implementação de Web Service no supervisório, não foi necessário fazer a integração dele com a camada MES através de OPC. Desta forma, houve uma economia de *tag* no supervisório, além de não ter sido necessária a implementação de um protocolo (sobre o protocolo OPC) para a troca de mensagens através dos *tags*, o que seria custoso, pois dificultaria manutenções futuras e seria um protocolo específico para cada fábrica. Os dados trocados entre duas camadas foram sendo, através de XML, um padrão aberto, auto-documentado e reaproveitável.

Durante o desenvolvimento foi questionado se a utilização de Web Services, para comunicação entre o MES e o supervisório, comprometeria a performance do sistema. Foram, então, realizados testes antes da implementação, que são mostrados a seguir:

Primeiramente, foi feito um teste para verificar o tempo gasto pelo cliente (consumidor do serviço) para receber o XML, processar o mesmo (fazer o parse) e alocá-lo em memória.

A máquina cliente utilizada para o teste tem a seguinte configuração: AMD Sempron 2800+/2.0GHZ, 512MB Ram – Win XP Professional SP2, JVM (Java Virtual Machine) 1.5.0-06. Esta máquina, daqui em diante, será chamada de Máquina 1.

Foram consumidas mensagens de forma crescente em relação ao seu tamanho. O gráfico abaixo mostra o resultado obtido:



Fonte: ATAN Sistemas

Figura 2. Tempo gasto para processar o XML

No eixo X, é mostrado o número de nós do XML consumido, enquanto o eixo Y mostra o tempo gasto para processamento em milisegundos.

O teste nos revelou que o tempo é viável para a operação do sistema, visto que a faixa de operação média (tamanho das mensagens trocadas, em número de nós) é em torno de 10 a 100 nós.

Além disto, o parser utilizado foi o DOM (Document Object Model), que é o padrão mais fácil de utilizar. No entanto, seu algoritmo tem um comportamento polinomial, considerando o tempo gasto em função do número de nós. Se fosse necessária a troca de mensagens maiores, poderia-se utilizar, por exemplo, o SAXParser.

Como os dados de produção (enviados do supervisório para o MES) e os dados de paradas (enviados da máquina onde está o servidor do PIMS), não podem ser “perdidos” em hipótese alguma – o que poderia ocorrer se o servidor do MES ficasse indisponível ou ocorresse uma falha de timeout – foi implementada uma fila de mensagens acoplada às máquinas geradoras destas mensagens. A troca de mensagens entre os softwares gerados, o iFix (no caso do supervisório) e o MES, é feita através de um Web Service local.

De forma a mostrar a confiabilidade no módulo de filas de mensagens, foram feitos dois testes distintos.

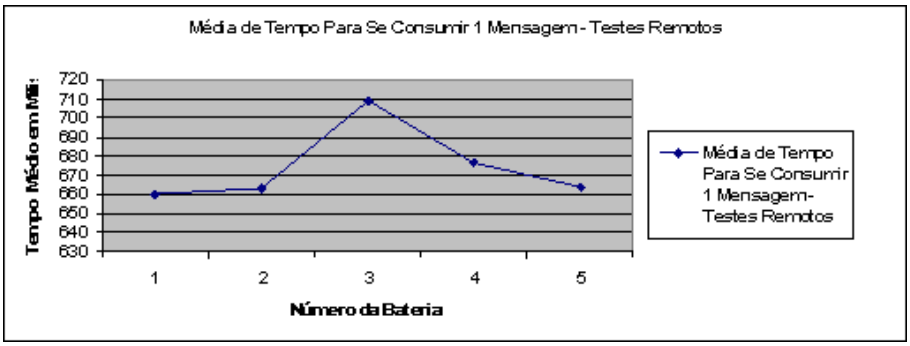
O objetivo do primeiro teste foi a medição do tempo médio de leitura das mensagens presentes na fila por parte do MES.

Foram feitas cinco baterias de consumo de 500 mensagens cujo tamanho das mensagens era de 28 caracteres.

A máquina que retirava as mensagens da fila, ou seja, a máquina consumidora, tem a seguinte configuração: Pentium 4 - 1.5 GHZ, 256 MB Ram - Win 2000 SP4, JVM (Java Virtual Machine) 1.5.0. Esta máquina, daqui em diante, será chamada de Máquina 2. A rede utilizada para teste foi uma rede corporativa de 100 MBits/s com cerca de 200 máquinas conectadas.

O tempo médio geral de leitura de cada mensagem, considerando as cinco baterias, foi de 674 milisegundos.

O gráfico abaixo mostra o resultado médio das cinco baterias.



Fonte: ATAN Sistemas

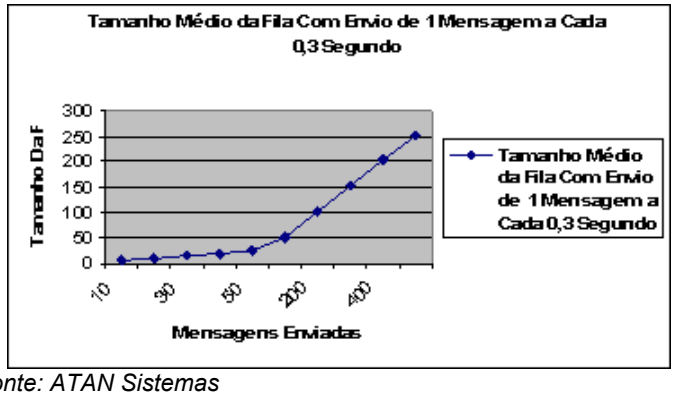
Figura 3. Média de tempo para se consumir uma mensagem

Pelo gráfico, pode-se concluir que a implementação é completamente viável, visto que o intervalo médio de envio de mensagens para a fila é muito maior que 1 (um) segundo.

O objetivo do segundo teste foi a medição do tamanho médio da fila para diferentes freqüências de envio de mensagens para a fila.

A máquina que enviava as mensagens para a fila era a Máquina 1, enquanto a máquina que retirava as mensagens da fila (máquina consumidora), era a Máquina 2, na mesma rede citada no primeiro teste. O tamanho da mensagem trocada era de 28 caracteres.

O resultado do envio é mostrado abaixo:



Fonte: ATAN Sistemas

Figura 4. Tamanho médio da fita com envio de uma mensagem a cada 0,3 segundos

Através dos testes, pode-se concluir que, apenas no envio com freqüência constante e bastante alta (em torno de uma mensagem a cada 400 milisegundos), é que uma fila substancial e crescente começa a ser formada.

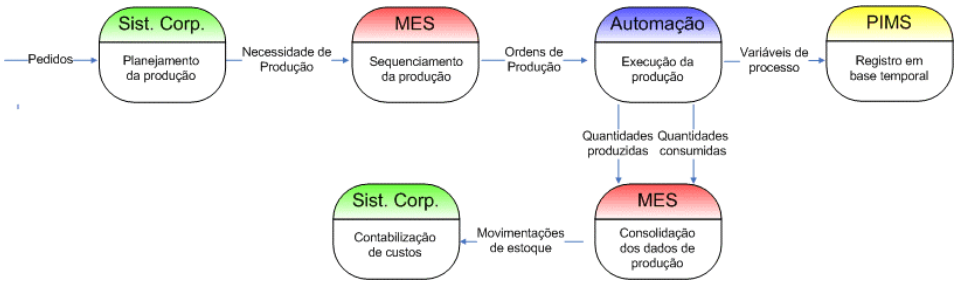
Desta forma, verifica-se que é viável a implementação do sistema, visto que o envio de mensagens tem uma freqüência bem menor que a freqüência crítica.

O que pode ocorrer é que, em momentos de pico, em que por instantes a freqüência for aumentada, será formada uma pequena fila, que se dissipará quando a freqüência baixar.

Apesar do uso de Web Services estar se tornando cada vez mais freqüente na área de TI, sua utilização em sistemas de automação ainda é rara, sendo por isso também um dos objetivos deste artigo exemplificar a utilização dos Web Services em Tecnologias de Automação (TA).

Usualmente, o nível de supervisão precisa buscar as informações da programação de produção e formulação dos produtos para executar, de forma correta, a produção. Estes serviços foram então disponibilizados para serem acessados pelo nível de supervisão. No supervisório, foi desenvolvido um script que acessa os serviços e interpreta o XML retornado. Os serviços que disponibilizam a programação de produção e formulação dos produtos poderão ser reutilizados por outros sistemas de controle ou por outros sistemas corporativos. Com o reuso do serviço diminuímos, mais uma vez o custo no desenvolvimento das interfaces.

A figura a seguir exemplifica a colaboração entre os sistemas ao longo do processo.



Fonte: ATAN Sistemas

Figura 5. Colaboração dos sistemas ao longo do processo

A tendência é que, no futuro, todas as informações já estejam disponibilizadas como serviços, bastando que sejam acessadas.

RESULTADOS ALCANÇADOS

As vantagens encontradas com a adoção desta arquitetura foram as seguintes:

- centralização dos dados: antes, quando os dados eram replicados em vários sistemas, havia mais chance de ocorrerem inconsistências. Para minimizar estas inconsistências, tornava-se necessário um esforço adicional para implementar rotinas de validação e sincronismo dos dados. A centralização em um único serviço facilita também a manutenção dos dados, que precisam ser alterados em apenas um lugar.
- centralização das regras de negócio: quando as interfaces são construídas como uma consulta ao banco de dados ou a um arquivo texto, as regras de negócio têm que ser reescritas por quem desenvolveu a interface. No caso dos serviços, as regras de negócio já estão no próprio serviço, o que evita replicação de código, facilitando a manutenção.
- padronização para toda a empresa: informações como a programação de produção ficam encapsuladas no serviço e, portanto, são padronizadas para todas as áreas da fábrica, enquanto o que é específico de cada linha de produção, como motores que devem ser acionados, método de controle das malhas, etc, é tratado de forma particular pelo supervisório de cada área.
- redução de custo nos próximos desenvolvimentos: a criação de uma base de serviços diminui consideravelmente os custos de desenvolvimento de interface para novas aplicações, porque as interfaces desenvolvidas serão reutilizadas.Uma outra economia encontrada é no número de *tags* utilizado pelo supervisório, pois, através desta arquitetura, não é necessário usar *tag* OPC para fazer a integração do supervisório com os outros sistemas.

PRÓXIMOS PASSOS

-Avançar na arquitetura orientada a serviço, desenhando os processos de outras áreas, organizando serviços disponíveis em um barramento e implementando serviços de alto nível para a utilização de uma ferramenta de BPM.

BIBLIOGRAFIA

1 Carreri, André Service-Oriented Architecture (SOA) agosto -2004. Disponível em: http://www.linhadecodigo.com.br/artigos.asp?id_ac=434&sub=0 Acesso em: jan. 2006
2 Endrei, Mark, Patterns: Service Oriented Architecture and Web Service abril 2004 - IBM
3 Gupta Samudra Service Oriented Architectura –Part 1. Disponível em: http://javaboutique.internet.com/tutorials/serv_orient/ Acesso em: março. 2006
4 Khan, Rashi. Understand Processes for SOA P.1 – 2 out 2005
5 Khan, Rashid SOA e BPM: duas siglas que caminham juntas. P1-3 set 2005
6 Repullo, Rodney Antonio – Qual o custo de não integrar os processos de Negócio. Disponível em: http://thebpmexperience.wordpress.com/ Acesso em: abril. 2006
7 Taurion, Cezar Entendendo os Web-service p-1 -3 jun-2005
8 Taurion, Cezar SOA: dando os primeiros passo p 1-3 dez -2004