SOA: CONCEITOS E TECNOLOGIAS UTILIZADAS NA ARQUITETURA ORIENTADA A SERVIÇO

SOA: concepts and technologies used in Service Oriented Architeture

André Luiz Casetto Vieira DA CRUZ
Faculdade Politécnica de Campinas

Peter JANDL JR
Faculdade de Jaguariúna
Faculdade Politécnica de Campinas

Resumo: Atualmente as empresas têm um conjunto de sistemas que necessitam estar integrado para manterem-se atualizados uns com os outros. O investimento em sistemas antigos é arriscado e caro e já estão estáveis e confiáveis. Mesmo com a aplicação de componentização e reutilização de implementações e componentes, há limitações e problemas de incompatibilidade de plataformas, com a necessidade de reutilização de processos de negócios já existentes entre sistemas cujas plataformas são heterogêneas, portanto não diretamente possível. Emergiram alternativas tais como o paradigma de orientação a serviço, o SOA é um conceito de Arquitetura Orientada a Serviço que está sendo muito aplicada em todo o mercado de TI, porem não se trata apenas de um conceito ou uma arquitetura utilizada para integrar as aplicações, as tecnologias utilizadas para aplicar possuem ferramentas com as funcionalidades para monitorar, gerenciar os processos de negócio da empresa através de modelagem dos processos de negócio conforme seu funcionamento, entre outros recursos.
Palavras-chave: SOA, arquitetura, orientada a serviço.

Abstract: Today the companies have a set of integrated systems that need to be updated to remain up to date with each others. Investment in old systems is risky and expensive and are already stable and reliable. Even with the implementation of components and reuse of implementations, there are limitations and problems of incompatible platforms, with the necessity for reuse of business processes from existing systems whose platforms are heterogeneous and therefore not directly possible. Emerging alternatives such as paradigm to guide service, the SOA is a concept of Service Oriented Architecture that is being applied across much of the IT market, but it is not just a concept or an architecture used to integrate applications, the technologies used to implement have own tools with the functionality to monitor, manage the company's business processes through modeling of business processes as their operation, among other resources.
Keywords: SOA; architecture; service oriented.

INTRODUÇÃO

Este artigo tem por objetivo explicar o conceito de *Service Oriented Architecture* (SOA) ou Arquitetura Orientada a Serviço. Também pretende apresentar os benefícios oferecidos, bem como os problemas e dificuldades para implantação do SOA. Para isso serão discutidos os termos conceituais utilizados, a divisão de camadas aplicadas com essa metodologia e as tecnologias que compõem o conceito, além dos termos utilizados para construção e implantação de SOA, tais como *Java Business Integration* (JBI), conceito de *Enterprise Service Bus* (ESB) e *Web Services*, e *Business Process Execution Language* (BPEL) para gerenciamento de processos.

Entendendo esses novos termos na área de arquitetura e desenvolvimento de sistemas, comentaremos sobre as ferramentas NetBeans e Open-ESB para desenvolvimento de SOA, que facilita a implementação e monitoração dessas tecnologias e aplicação dos conceitos.

VISÃO DO CONCEITO DE SOA

SOA, *Service Oriented Architecture* (Arquitetura Orientada a Serviço), é o mais novo padrão utilizado em arquiteturas e desenvolvimento de sistemas para a realização de alinhamento de Tecnologia de Informação (TI) com estratégias organizacionais para modelar, automatizar e monitorar processos de negócio. Trata-se de um meio de integrar diversos sistemas (de um lado a outro) utilizando protocolos padronizados e interfaces convencionais – normalmente serviço de *web*, para facilitar o acesso à lógica e informação empresarial entre diversos serviços (Accenture, 2008).

SOA fornece os padrões a serem seguidos e a orientação necessária para transformar uma matriz existente de uma empresa de recursos de TI que seja heterogênea, distribuída, complexa e inflexível em recursos integrados, simplificados e de alta flexibilidade que podem ser mudados e compostos pelas regras de negócios mais diretamente afetadas.

SOA, em última análise, favorece a entrega de uma nova geração de aplicações dinâmicas (às vezes chamado de aplicações compostas). Essas

aplicações fornecem aos usuários finais as mais precisas e completas informações e dicas sobre o processo, assim como a flexibilidade para integrá-los da maneira mais adequada e acessá-los por meio da interface de apresentação, seja através de uma interface *web*, ou de uma aplicação cliente, ou mesmo em uma aplicação em dispositivo móvel (Accenture, 2008; MSDN, 2008).

Além disso, SOA representa um modelo arquitetural que está posicionado no centro da plataforma de serviços orientada à computação e sustentada pela aplicação do paradigma de design orientado a serviço. Em uma abordagem arquitetural corporativa é possível a criação de serviços de negócio interoperáveis que podem facilmente ser reutilizados e compartilhados entre aplicações e empresas (Arsanjani, 2008).

O conceito do serviço não é novidade, mas a noção de SOA tem se transformado ao longo dos últimos dois anos. Trata-se de um estilo arquitetônico de construção de aplicações de software que promove baixo acoplamento entre componentes, de modo que possam ser reutilizados. Assim sendo, é uma nova forma de construir aplicações com as seguintes características (Accenture, 2008):

- Serviços são componentes que publicam contratos/interfaces; esses contratos são sistemas independentes de plataforma, linguagem e operação.
- Consumidores podem descobrir os serviços de maneiras dinâmicas.
- Os serviços são interoperáveis, de modo a se comunicar de forma transparente entre os sistemas, trabalhando com padrões abertos.

O elemento básico de SOA é o serviço, considerado como um módulo de software autocontido que executa uma tarefa predeterminada. Na verdade são componentes de software que não requerem que os desenvolvedores utilizem uma tecnologia subjacente especifica. Assim desenvolvedores Java, tem a tendência de focar na reutilização do código, e de integrar firmemente a lógica do objeto ou dos componentes em uma aplicação. No entanto, o SOA

promove a aplicação de montagem, porque os serviços podem ser reutilizados por inúmeros consumidores (Gopalan,et al., 2008).

Além disso, SOA também permite automatizar a gestão do processo empresarial que pode consumir e harmonizar esses serviços para alcançar a funcionalidade desejada. Assim, novos processos empresariais podem ser construídos através da utilização de serviços pré-existentes.

BENEFÍCIOS

A Arquitetura SOA é vista como um passo adiante na evolução no que concerne ajudar organizações de TI a encontrarem mais rapidamente desenvolvimento de aplicações, TI de menor custo, e maior adaptabilidade às necessidades empresariais em constante mudança. Além disso, devido aos significantes benefícios que pode oferecer, SOA está rapidamente se tornando um caminho dominante para a implementação de serviços de TI (Accenture, 2008).

SOA também colabora para as organizações transformarem seus processos empresariais mais facilmente, garantindo maior performance através da simplificação dos sistemas de informação subjacentes.

Abordagens arquitetônicas antigas, que já expandiram as oportunidades empresariais, agora limitam o crescimento – porém, os sistemas existentes não podem ser simplesmente substituídos. Dessa forma SOA oferece a esses sistemas a flexibilidade e agilidade para responder a um ambiente empresarial que está mudando rapidamente.

SOA permite às empresas e governos oportunidade de capitalizar sobre (Accenture, 2008):

> **Maior agilidade**: permitindo que as organizações respondam rapidamente aos novos imperativos comerciais, desenvolvendo capacidades distintas e serviços existentes de alavancar para respostas verdadeiras. Negócios e TI estão mais proximamente alinhados.

Redução de custos: promovendo a reutilização de ativos existentes, aumentando sua eficiência e reduzindo os custos de desenvolvimento de aplicações. Também permite que sistemas de TI alavanquem rapidamente as bases de código e serviços mais prontamente disponíveis de toda e qualquer organização. Além disso, melhora a coordenação entre vendas, marketing, distribuição e canais de parceria, reduz custos e soluciona problemas de consumo de tempo.

Retorno sobre o investimento (*Return On Investment* ou ROI): enquanto a arquitetura SOA oferece uma base de alta performance, valor e o retorno de investimento encontram-se no projeto que permite aplicar SOA. Escolhe projetos que oferecem ROI em seus próprios termos, ou aqueles que são obrigados a oferecer ganhos competitivos ou novas capacidades.

SOA também permite às organizações alcançar suas metas de TI. O valor tecnológico com SOA inclui:

Sistemas mais simples: baseando-se em padrões industriais é possível reduzir sua complexidade quando comparado a sistemas integrados em uma base solução-pós-solução. Também permite aplicações futuras para criar uma malha em perfeita harmonia com serviços baseados em padrões existentes.

Custos de manutenção: a simplicidade e facilidade de manutenção significam que custos de manutenção são reduzidos e a valiosa equipe de TI é liberada para trabalho estratégico.

Reforço da flexibilidade arquitetônica: oferece apoio para a construção de nova geração de soluções compostas. Essas soluções de performance conduzida consolidam inúmeros processos empresariais de vários sistemas em uma simples interface de usuário.

Redução de custos de integração: torna possível de serem desenvolvidos, implementados e reutilizados por organizações, processos que são tecnicamente habilitados e integrados através do uso

de padrões de Web Services como o *eXtensive Markup Language* (XML), *Simple Object Access Protocol* (SOAP) e *Web Services Description Language* (WSDL). Além disso, a conectividade, a transferências de dados e esforços para integração de processos são simplificados, reduzindo integração e desenvolvimento relacionados com os custos de suporte.

Melhora na agilidade empresarial: permite, além de um desenvolvimento mais rápido, a implantação de novos serviços, permitindo que as organizações de TI montem rapidamente um negócio de nível superior de serviços a partir de componentes reutilizáveis de software. Estes serviços empresariais também podem ser rapidamente alterados, tornando mais fácil a mudança nos processos empresariais. Como resultado, serviços de TI podem ser mais alinhados com as necessidades do negócio.

Melhora na eficiência operacional: resulta em serviços de TI mais adaptáveis, bem como em uma melhor visibilidade e capacidade de adaptação dos processos de negócio que são assistidos por esses serviços. Essa flexibilidade e visibilidade tornarão mais fácil integrar os *workflows* através de múltiplas funções empresariais e otimizar processos de negócios com maior eficiência. Isso pode resultar em maior produtividade para os trabalhadores e operações mais eficientes com os clientes.

Redução dos serviços de TI: componentes de software reutilizáveis em um sistema de SOA também podem reduzir o tempo e custo do desenvolvimento e manutenção do software. SOA pode eliminar a necessidade de integração ponto-a-ponto entre aplicações, reduzindo enormemente o custo e a complexidade da posição estratégica e manutenção do software. Investimentos em sistemas legados também podem ser protegidos pela exposição das funcionalidades neles existentes, tornando-os serviços web que podem, então, ser usados por outras aplicações.

PROBLEMAS E DIFICULDADES PARA IMPLANTAR

O uso de tecnologias e aplicações heterogêneas em corporações é uma realidade. Numa era em que os recursos são escassos, lojas de TI não podem simplesmente descartar suas aplicações existentes; em vez disso, devem alavancar seus investimentos prévios. SOA é popular porque permite a reutilização de aplicações e promete a operação conjunta entre aplicações e tecnologias heterogêneas. Mas esse conceito se defronta com barreiras técnicas e recursos profissionais.

Problemas comuns são: o desconhecimento da complexidade de SOA é a criação e gerenciamento software; poucos profissionais estão capacitados para desenvolverem projetos utilizando a arquitetura SOA e as tecnologias utilizadas; além de resistências as mudanças, pois como SOA produz mudanças gigantescas em uma organização, o resultado é óbvio.

No modelo ESB, um fornecedor precisa implementar muitos algoritmos de transformação e deve entender como proceder com diversos formatos tais como: COBOL *copybook*, MDDL, XBRL, RIXM, etc., o que não é uma tarefa fácil. Por outro lado, todas as mensagens que trafegam dentro do BUS (NMR) devem ser convertidas para XML mesmo se o *end point* receptor possa processar a mensagem no formato original; com isso causando um processamento adicional sobre o servidor de integração. Dessa maneira, o integrador, não precisa se preocupar sobre formatos diferentes e sua conversão. Mas o fornecedor precisa implementar muitos mecanismos para cobrir todos os requisitos personalizados. Outra questão é a de que o integrador é responsável por converter mensagens no formato disponível para o formato preciso, da maneira que achar mais adequado para seu trabalho e conteúdo semântico da mensagem (OPEN-ESB, 2008).

PRINCÍPIOS E ESTILOS DA ARQUITETURA

O estilo de arquitetura definida em SOA descreve um conjunto de padrões e formatações para criar baixo acoplamento e serviços alinhados a negócio que, devido à separação de conceitos entre descrições,

implementações e caráter vinculativo, proporciona flexibilidade sem precedentes em respostas às novas ameaças e oportunidades empresariais.

SOA é uma escala empresarial de arquitetura de TI para interligar recursos sob demanda. Os recursos são disponibilizados aos participantes em linhas de negócios (normalmente abrangendo múltiplas aplicações dentro de uma empresa ou por várias empresas). Consiste de um conjunto de serviços de TI alinhados à atividade das empresas que cumprem um conjunto de processos e metas de negócios da organização. Você pode coreografar estes serviços em aplicações compostas e invocá-los através de protocolos padronizados (Arsanjani, 2008).

O serviço é um recurso de software (detectável), com uma descrição do serviço exteriorizado. Essa descrição do serviço está disponível para pesquisa, *binding* podendo ser invocado por um consumidor dos serviços.

Agilidade de negócios é adquirida por sistemas de TI que são flexíveis, primeiramente por separação das interfaces, implementações, e *binding* (protocolos) oferecidos por SOA, podendo reutilizar os serviços através de unidades internas de negócio ou através de cadeia de valores entre os parceiros comerciais em um modelo de realização fracionada. A Realização fracionada refere-se à habilidade de um estilo arquitetônico de aplicar seus padrões e suas regras, associadas a seus participantes nesse modelo de interação de maneira composta, podendo ser aplicado em uma ou múltiplas camadas através da arquitetura empresarial. No caso dos projetos, isso pode ocorrer entre unidades e parceiros de negócio dentro de uma cadeia de valores em um modelo conceitual de maneira escalar.

CAMADAS

Em uma visão abstrata a arquitetura SOA é retratada como uma arquitetura particionada em várias camadas compostas por serviços que são alinhados com os processos de negócio (Arsanjani, 2008).

Os relacionamentos entre serviços e componentes estão ligados às escalas empresariais dos componentes (granularidade empresarial ou linha de

componentes de negócio) realizando os serviços e provendo suas funcionalidades e mantendo a qualidade dos serviços. Fluxos de processos podem ser sustentados por uma coreografia dos serviços expostos nas aplicações compostas. Uma arquitetura integrada suporta o roteamento, mediação e a transformação de dados desses serviços, componentes e fluxos usando um *Enterprise Service Bus* (ESB). O serviço implantado deve ser monitorado e gerenciado pela qualidade dos serviços e aderência para requisitos não funcionais (Accenture, 2008).

Para cada uma das camadas, deve ser feito o design e as decisões arquiteturais. Por isso, na documentação de SOA deve ser criado um documento constituído por seções que correspondem a cada uma das camadas (Figura 1), a saber (Arsanjani, 2008):

- Nível 1: Sistemas Operacionais;
- Nível 2:Componentes Empresariais;
- Nível 3:Serviços;
- Nível 4:Composição de processos de negócio ou Coreografias;
- Nível 5:Acessos ou Apresentação;
- Nível 6:Integração; e
- Nível 7: Qualidade de Serviços.

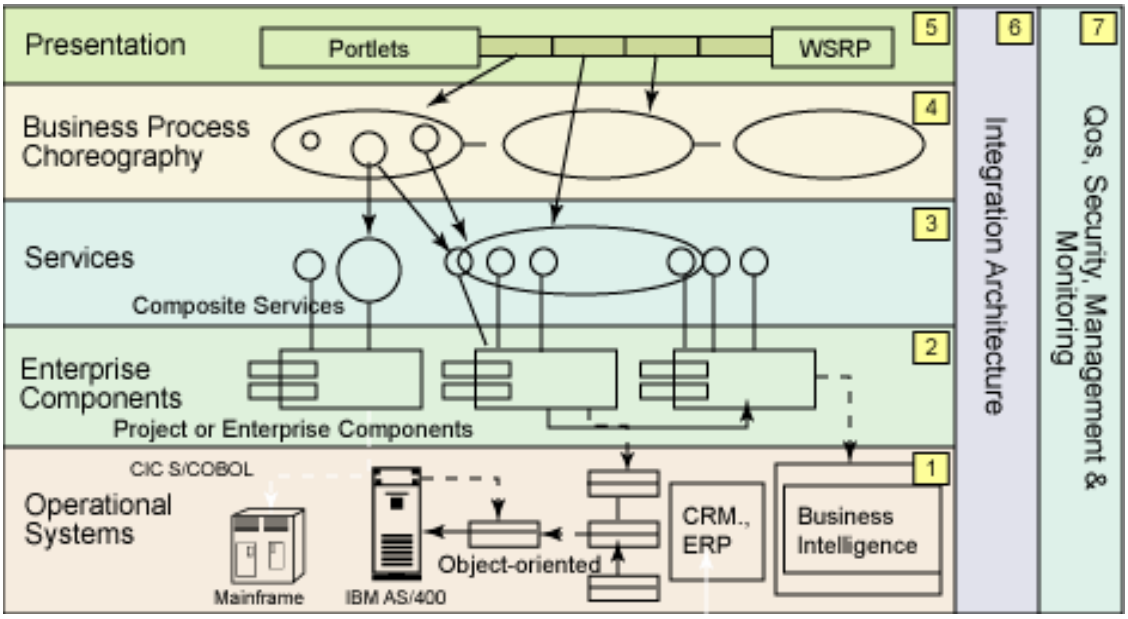

Figura 1. Camadas arquiteturais.

Nível 1: Camada dos Sistemas Operacionais. Essa camada consiste em aplicações personalizadas já existentes, ou, também chamados, sistemas legados incluindo os já existentes pacotes de aplicação CRM e ERP, e alguns sistemas antigos implementado com orientação a objeto, tão bom quanto aplicações com negócios inteligentes. A composição de camadas de arquitetura em SOA pode alavancar sistemas existentes e integrá-los usando técnicas de integração orientada a serviços.

Nível 2: Camada de componentes empresariais. Essa camada contém os componentes que são responsáveis por realizar a funcionalidade e manter a Qualidade dos Serviços (QoS) expostos. Esses componentes são gerenciados e regidos por um conjunto de ativos empresariais que são financiados pela empresa ou pelo nível da unidade do processo. Essa camada costuma utilizar tecnologias de container assim como servidores de aplicação para implementar os componentes, gerenciando carga de trabalho, alta disponibilidade e balanceamento de carga.

Nível 3: Camada de Serviços. Os serviços de negócio são expostos nessa camada. Podem ser encontrados ou vinculados estaticamente e então invocados, ou, coreografados em uma composição de serviços, que prove o mecanismo para escalar componentes empresariais e de unidades de negócio especifica, e em alguns casos, componentes de especificação de projetos, e externaliza um conjunto dessas interfaces no formado de serviços descritos. Assim, os componentes empresariais realizam serviços utilizando a funcionalidade de execução provida pelas suas interfaces. Essas interfaces são expostas como descrições de serviço nessa camada, onde estão disponibilizadas para uso, podendo existir isoladamente ou como uma composição de serviços.

Nível 4: Camada de Composição de processos de negócio ou Coreografia. Composições e coreografias de exposição de serviços da Camada 3 são definidas nessa camada. Serviços são agrupados em um fluxo através da orquestração ou coreografia, portanto, agem em conjunto como uma única aplicação. Essas aplicações suportam específicos casos de uso e processos de negócio.

Nível 5: Camada de Apresentação. Embora essa camada normalmente esteja fora do escopo de discussão sobre SOA, ele está se tornando cada vez mais relevante.

Nível 6: Integração (ESB). Essa camada permite a integração de serviços através da introdução de um conjunto confiável de capacidades, como roteamento inteligente, mediação de protocolo, e outros mecanismos de transformação, muitas vezes descrito como o ESB. *Web Services Description Language* (WSDL) especifica o bi*n*ding, o qual define o local onde o serviço está disponibilizado, como acessá-lo e quais as operações ou métodos disponíveis. Por outro lado, um ESB provê uma localização independente do mecanismo para integração.

Nível 7: Qualidade de serviço (QoS). Essa camada fornece a capacidade de monitorar requisições, gerenciar e manter QoS assim como segurança, performance e disponibilidade. Este é um processo background através dos mecanismos de requisição-e-respostas e ferramentas que monitoram as aplicações SOA, incluindo todos os padrões importantes para a implementação do WS-Management e outros protocolos e padrões que implementam a qualidade dos serviços para SOA.

WEB SERVICES

Web Service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os *Web Services* são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria "linguagem" que é traduzida para uma linguagem universal, o formato XML.

As bases para a construção de um *Web Service* são os padrões XML e SOAP, como mostra a Figura 2. O transporte dos dados é realizado normalmente via protocolo HTTP ou HTTPS para conexões seguras (o padrão

não determina o protocolo de transporte). Os dados são transferidos no formato XML, encapsulados pelo protocolo SOAP, os serviços (operações, mensagens, parâmetros, etc.) são descritos usando a linguagem WSDL; o processo de publicação/pesquisa/descoberta de *Web Services* utiliza o protocolo UDDI (*Universal Description, Discovery and Integration*).

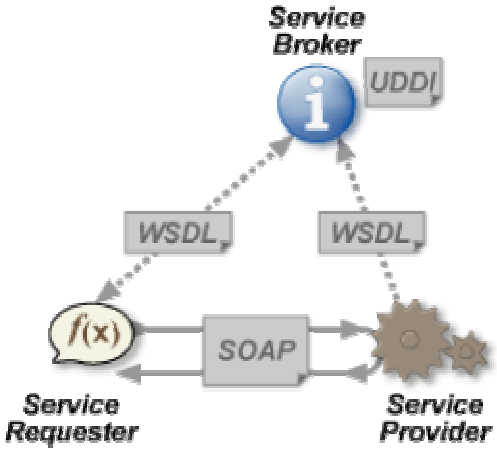

Figura 2. Padrões *Web Services* são as interfaces que permitem que máquinas e aplicações se comuniquem. UDDI permite que clientes encontrem *Web Services*.

WEB SERVICES E SOA

Primeiramente vamos diferenciar SOA de *Web Services*. SOA é uma estratégia global para desenvolvimento de aplicações de software dentro de uma companhia, conhecida como programação orientada a serviço. Web services é um conjunto de tecnologias, incluindo XML, *Simple Object Access Protocol* (SOAP), *Web Services Description Language* (WSDL) e *Universal Description, Discover and Integration* (UDDI), que permitem a construção de soluções de problemas de programação para um meio específico de transmissão de mensagens e de integração de aplicação.

Web Services e SOA tratam-se de concepção e construção de sistemas a partir de uma rede "endereçável" de componentes de software heterogêneo. Essa combinação resolve as questões da abordagem de CORBA e DCOM ao SOA. *Web Services* removeram outra barreira com a concessão da interconexão de aplicações de uma maneira neutra com modelo de objetos. Por exemplo, utilizando um simples esquema de transmissão de mensagens

com base XML; aplicações Java podem invocar aplicações Microsoft.NET, CORBA, ou até COBOL e a aplicação invocada não precisa saber onde a transação vai ocorrer, em qual linguagem é escrita, ou, qual via a mensagem tomará ao longo do caminho. Um serviço é requerido, e uma resposta é fornecida. *Web Services* é um conjunto de tecnologias necessárias para o SOA, e o SOA está se tornando a arquitetura de escolha para o desenvolvimento de aplicações com respostas positivas e adaptáveis.

O sucesso dos vários *Web Services* mostrou que a tecnologia existe e pode permitir a implementação de um verdadeiro SOA, podendo ser tanto uma arquitetura como um modelo de programa, uma maneira de pensar sobre a construção de software. SOA permite projetar sistemas que oferecem serviços a outras aplicações através das interfaces editadas e possíveis de serem encontradas, e onde os serviços podem ser invocados sobre uma rede. Quando se implementa um SOA utilizando tecnologias de *Web Services*, cria-se uma nova maneira de construir aplicações com um modelo de programa mais poderoso e flexível. É possível reduzir seu custo de desenvolvimento e o risco de implementação.

ESB

Um ESB (*Enterprise Service Bus*) geralmente provê uma camada de abstração acima de um sistema de mensagens corporativo, que permite aos arquitetos de integração explorar todas as possibilidades e benefícios deste sistema de mensagem sem a necessidade de escrever código. Os fundamentos do ESB são baseados na decomposição de processos de negócio executando de forma harmoniosa (KALALI, 2008).

Um ESB traz conceitos relacionados com o fluxo, como a transformação e encaminhamento para SOA. Também pode prover uma abstração de *endpoints*. Isso promove flexibilidade na camada de transporte e habilita acoplar serviços POJO.

A abstração de *endpoints* consiste basicamente nos serviços se registrarem em um ponto único, que é o barramento, onde vários clientes são

atendidos por serviços que podem trabalhar isolados ou em composição para atender uma demanda de negócio.

No mundo ESB os serviços não interagem diretamente um com o outro. De preferência, o executor ESB atua como um mediador entre os serviços para haver o baixo acoplamento entre eles. O executor ESB pode implementar protocolo *bindings*, tradutor de mensagem, manipulação de mensagem, etc.

Principais funcionalidades realizadas pelo ESB incluem:

- Invocação: Protocolos de transporte Síncrono e Assíncrono, serviço de mapeamento (localização e *binding*);
- Roteamento: Endereçamento, roteamento estático/determinístico, roteamento baseado em conteúdo, roteamento baseado em policiamento.
- Mediação: Adaptador, transformação de protocolos, mapeamento de serviço.
- Mensagens: Processamento de mensagem, transformação de mensagem e acessório de mensagens.
- Coreografia: Implementação de processos de negócios complexos
- Orquestração: Coordenação de múltiplas implementações de serviços expostos com somente um, agregação de serviços.
- Processamento de Eventos Complexos: Interpretação de eventos, correlação.
- Outras qualidades de serviço: Segurança, entrega confiável, gerenciamento de transação.
- Gerenciamento: Monitoração, auditoria, *logging*

A maioria dos fornecedores ESB se baseia nas propostas dos padrões abertos de SOA e tecnologias incluindo os vários padrões *Web Services* e protocolos. Eles provêm uma variedade de *bindings* de transporte para invocar os serviços incluindo HTTP, FTP, and JMS etc. A maioria dos ESBs utiliza WS-BPEL (*Business Process Execution Language for Web Services*) para realizar orquestração entre os serviços implantados para implementar os processos de negócio. Fornecedores ESB também provêem qualidade das características de

serviço incluindo tolerância à falha, *failover*, balanceamento de carga, *buffer* de mensagens, etc.

WS-BPEL

Business Process Execution Language (BPEL) para *Web Services* é baseado em uma linguagem XML concebida para possibilitar o compartilhamento de tarefas para computação distribuída ou ambiente de computação em grade - mesmo através de várias organizações – utilizando uma combinação de *Web Services*. Criada por desenvolvedores da BEA Systems, IBM, e Microsoft, BPEL caminha e substitui IBM's WebServices Flow Language (WSFL) e as especificações Microsoft's XLANG. (BPEL às vezes também é identificado como BPELWS ou BPEL4WS.)

Processos de negócio tipicamente envolvem o intercâmbio, ou orquestração, de mensagens entre os processos e outros *Web Services* conhecidos como serviços parceiros. O contrato entre um processo de negócio e serviços parceiros é descrito em um WSDL 1.1. As mensagens trocadas entre processos de negócio e serviços parceiros são empacotadas, como definido pela especificação JBI, e roteado via JBI *Normalized Message Router* (NMR). O NMR interage com os *Web Services* externos, não residentes na JVM local, via *binding components*, os quais são responsáveis por encapsular detalhes de protocolos específicos. Transações entre o BPEL *Service Engine* e EJBs ou componentes web são tratadas pelo mecanismo de serviço do Java EE.

JBI

Java Business Integration (JBI) consiste em uma especificação que define uma arquitetura e comportamentos para hospedagem de componentes plugáveis, onde a comunicação entre estes componentes se fará através do uso de WSDL (*Web Services Description Language*) 2.0 (Barros, 2008, Vince, 2008).

O principal objetivo do JBI é permitir que possamos utilizar componentes plugáveis, provendo todas as funções necessárias para uma solução de

integração, sem sermos dependentes de um fornecedor específico. Ou seja, os componentes poderão ser facilmente portáveis entre quaisquer implementações do JBI.

O JBI possui dois componentes principais, os *Service Engines* e os *Binding Components*, como visto na Figura 3.

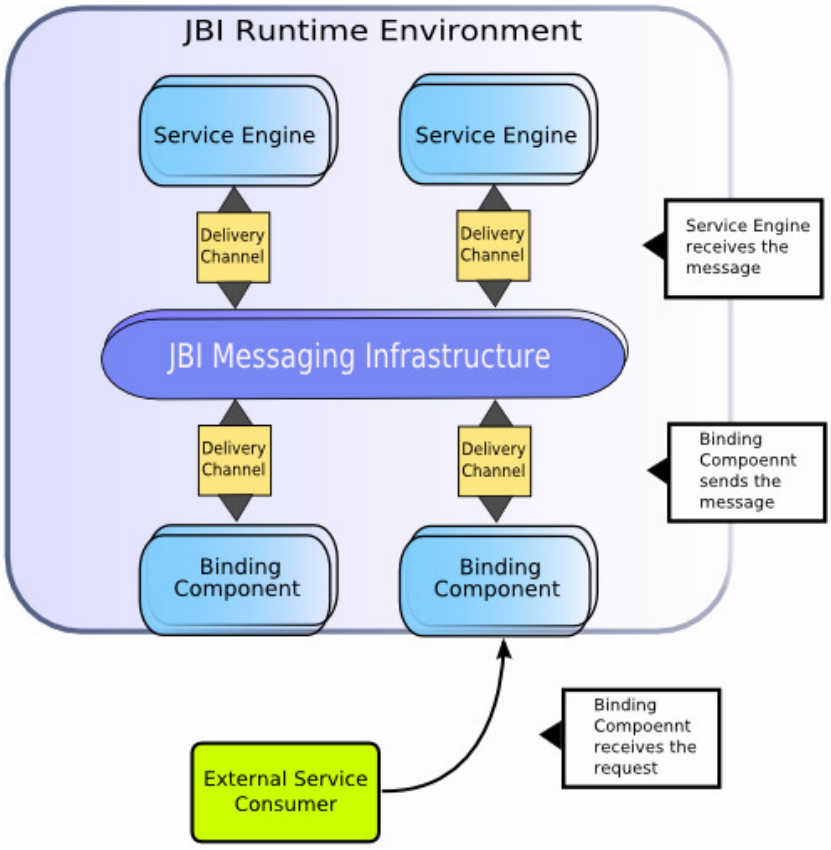

Figura 3. Componentes JBI (*Service Engines* e *Binding Components*) em um ambiente de execução.

Os *Service Engines* provêem ou consomem serviços localmente dentro do ambiente de execução JBI, e são responsáveis por prover lógicas de negócios como serviços, transformações de dados, e roteamento de serviços (Barros, 2008, Vince, 2008).

Binding Components provêem protocolos independentes para transporte ou comunicação. Eles acessam serviços remotos usando um protocolo especifico, e disponibilizam os serviços para o JBI *Normalized Message Router* (NMR). Desse modo outros componentes JBI podem acessar esses serviços no NMR (Barros, 2008, Vince, 2008).

Os *Binding Components* também são especializados em especificar protocolos externos, como HTTP, JMS, e outros. Permitindo que qualquer componente se comunique sobre qualquer protocolo ou transporte disponível pelo *Binding Components* implantado no ambiente JBI, como ilustrado pela Figura 4. Não havendo necessidade de implementar esses protocolos nas regras de negócio.

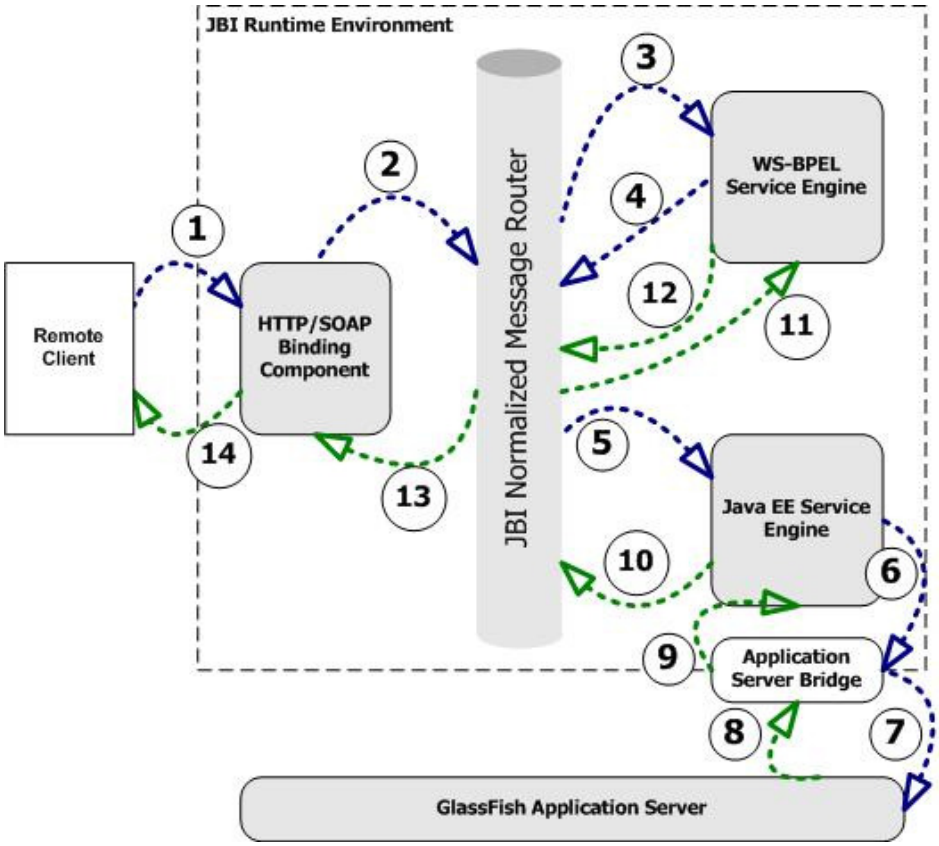

Figura 4. Fluxo de mensagem de um ambiente de execução JBI utilizando *Service Engines, Normalized Message Router* e *Binding Components.*

A distinção entre os *Service Engines* e os *Binding Components* é apenas de caráter arquitetural, separando as lógicas de negócios das lógicas de comunicação com o intuito de reduzir a complexidade de entendimento, implementação e flexibilidade.

OPEN-ESB

Open ESB é um *Enterprise Service Bus* desenvolvido em Java seguindo os padrões livres e em *open source*. Sua arquitetura pode ser vista na Figura 5.

É uma implementação *open source* do JBI apoiada pela Sun Microsystems e hospedada em Java.net, é um dos maiores ESB *open source* para integração, ou SOA. Open ESB pode ser utilizado nos servidores de aplicação JBoss, GlassFish e Websphere com a possibilidade de ser executado como uma aplicação Java SE. Open ESB é pré-empacotado com muitos *binding components* e *Service Engine* o qual quase deixa os desenvolvedores e integradores livres para procurar por *binding components* ou *service engines* que serão necessários nos cenários típicos de integração (Open-ESB, 2008).

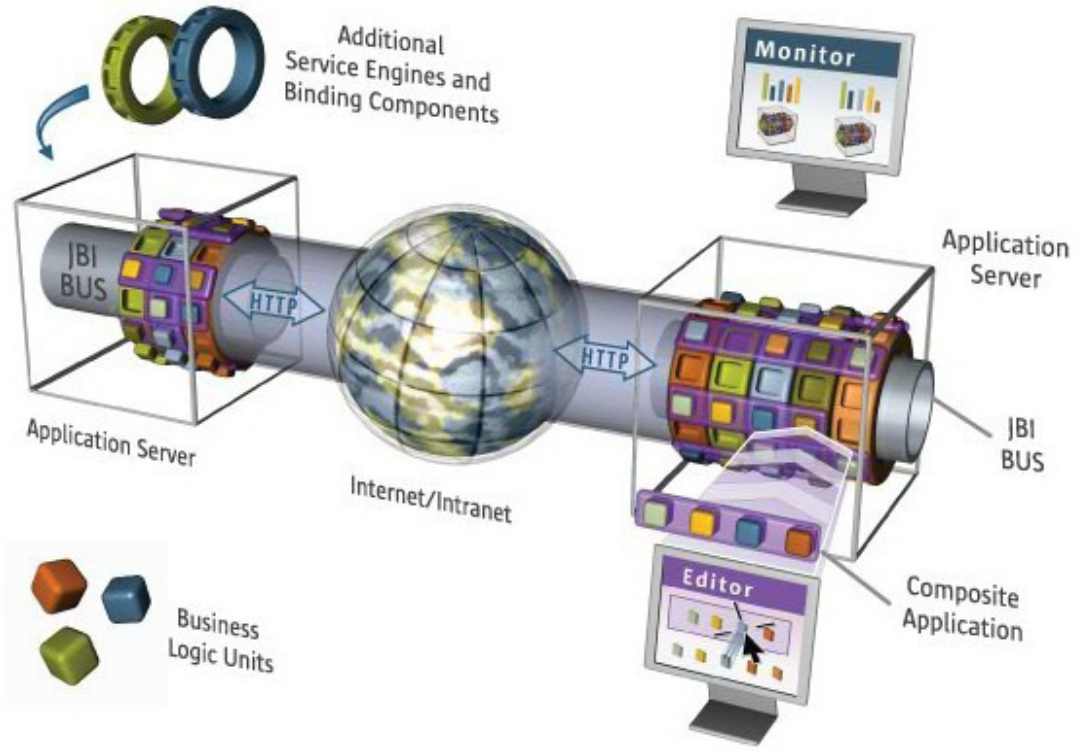

Figura 5. Arquitetura do Open-ESB.

O Open ESB pode ser utilizado como uma plataforma para *Enterprise Application Integration* (EAI) e SOA. Assim, o Open ESB contém vários componentes para transformação de dados, orquestração, e conectividade. Também oferece suporte para HTTP e *Web Services*, JMS, diversos bancos de dados relacionais, MQ Series, SAP, IMS, HL7, etc. Os processos lógicos de negócios podem ser expressos com BPEL, EJBs, POJOs. Também há um

suporte para processamento de eventos complexos. É possível consultar o projeto *web site* do Open-ESB para obter uma lista completa dos componentes.

Além da abundância de *Service Engines* e *Binding Components*, Open ESB está muito bem integrado a IDE de desenvolvimento NetBeans IDE 6.1 com muitos recursos que facilitam o desenvolvimento de aplicações compostas baseadas em Open ESB., tais como a ferramenta de projeto visual que permite "arrastar e largar", configuração de conectividade, configuração de *binding components*, elementos necessários para implementação de *Web Services*. Cada *binding component* no Open ESB tem uma representação de *design* de tempo na IDE, facilitando o design do cenário de integração, baseado nas facilidades do servidor.

CONSIDERAÇÕES FINAIS

O objetivo desse artigo foi pesquisar sobre Arquitetura Orientada a Serviço, quais os benefícios e problemas na aplicação dessa arquitetura. Esse estudo sobre SOA e os conceitos utilizados servem como primeiros passos para se aprofundar nas tecnologias e técnicas utilizadas, além de entender o porquê do uso de SOA no mercado. O intuído desse artigo foca a teoria, mas fica o convite de aprofundar no estudo das tecnologias e aplicar o uso delas, implementando *Web Services* e ESB gerenciando os fluxos das mensagens e mapeando-as com BPEL e monitorando os processo de negócio. Destacamos algumas ferramentas que facilitam a implementação de projetos que utilizam esse conceito de arquitetura.

REFERÊNCIAS BIBLIOGRÁFICAS

ACCENTURE. **Service-Oriented Architecture (SOA) Benefits, Advantages and Values**. Disponível em: http://www.accenture.com/Global/Technology/Service_oriented_Architecture/ServiceOrientedBenefits.htm. Acessado em 30/10/2008.

BARROS, Breno. **Entendendo o JBI**. Disponível em: http://soa-journal.blogspot.com/2007/07/entendendo-o-jbi.html. Acessado em 30/10/2008.

GOPALAN Suresh Raj, BINOD PG, KEITH Babo, PALKOVIC Rick. **Implementing Service-Oriented Architectures (SOA) with the Java EE 5 SDK**. Disponível em: https://open-esb.dev.java.net/public/whitepapers/ImplementingSOA.pdf. Acessado em 30/10/2008.

ARSANJANI, Ali. **Service-oriented modeling and architecture**. Disponível em: http://www.ibm.com/developerworks/library/ws-soa-design1/. Acessado em 30/10/2008.

KALALI, Masoud. **Introduction to integration and Open ESB, part I: Open ESB serverside artifacts**. Disponível em: http://java.dzone.com/articles/introduction-integration-and-o. Acessado em 30/10/2008.

MSDN. **Service Oriented Architecture (SOA)**. Disponível em: http://msdn.microsoft.com/en-us/library/bb833022.aspx. Acessado em 30/10/2008.

OPEN-ESB. **The open source ESB & SOA Integration**. Disponível em: http://wiki.open-esb.java.net/Wiki.jsp?page=LearnJBI. Acessado em 30/10/2008.

VINCE, Genovese. **JBI Component Technical Overview**. Disponível em: http://www.netbeans.org/kb/60/soa/jbi-tech-overview.html. Acessado em 30/10/2008.