

WEB SERVICES: MIDDLEWARES EM AMBIENTES HETEROGÊNEOS

Antônio Costa Neto, César Barbosa, Fernanda Michelini, Iara Montes Ferreira, Robson Pereira Mendonça, Thiago Grecco Rolli

Centro Universitário Fundação Santo André

RESUMO

A necessidade cada vez maior de sistemas que possam interagir entre si e de uma maneira rápida, eficiente e segura fez com que a tecnologia de *middlewares*, dentre eles, *Web services*, se tornasse um dos assuntos mais abordados e explorados por profissionais de sistemas de informação, ciência da computação e áreas afins, nos dias de hoje. A explosão/popularização da *World Wide Web* nos últimos anos foi, sem dúvida, o motor propulsor não somente de *Web services*, mas da Tecnologia da Informação como ciência. Esse crescimento identificou a necessidade de homogeneidade entre as diferentes aplicações e fez surgir os *middlewares*, os quais não são uma revolução, mas uma evolução no conceito de conectividade entre diferentes sistemas. Esse tipo de serviço vem desempenhando um papel de grande importância na integração de sistemas de informação na Internet, principalmente sistema do tipo B2B.

Palavras-chave: web services, sistemas integrados, *middleware*, tecnologia da informação.

ABSTRACT

The great necessity of interacting systems in a fast, efficient and safe way makes the middleware technology an important issue among professionals of computing science area. The WEB explosion in the latest years was a force of growing the *Web services* and the information technology as a science. That growing identified the needed of homogeneity between different applications and makes appear the middleware technology, which are not a revolution, but an evolution in the concept of connectivity between different systems. This type of services has an important role in the integration of information systems over the internet, mainly for B2B systems.

Keywords: web services, integrated systems, middleware, information technology.

mesmas funcionalidades é um grave problema presente em muitas empresas (BONIAT, 2005).

Considere, por exemplo, o seguinte cenário, tipicamente presente em um ambiente empresarial. Determinada empresa possui uma versão de software, baseada em aplicações executáveis, que é utilizada através de sua Intranet e outra versão, baseada em páginas dinâmicas, disponíveis para acesso externo, por meio da Internet. O fato do legado das empresas continuar a operar e, ao mesmo tempo, apresentar novas demandas, que, muitas vezes, sua tecnologia não suporta, ocasiona o surgimento de diferentes versões de software que implementam a mesma lógica de negócio. Acrescentando nesse cenário o compartilhamento de dados com sistemas portáteis, como WAP, muito provavelmente, seria criada uma terceira versão de software implementando as mesmas funcionalidades daquelas já existentes, porém agora sobre uma nova tecnologia.

Supondo a necessidade de alteração na lógica de qualquer uma das aplicações, para manter a compatibilidade, essa alteração deverá ser reescrita nas demais.

Dividir o software em camadas (dados, negócio, interface com o usuário) tem se mostrado uma das melhores alternativas para o desenvolvimento de aplicações dessa natureza. A presença de características heterogêneas nas aplicações e no processo de desenvolvimento aliado à grande quantidade de tecnologias existentes para implementação de software distribuído, porém, se constitui em um problema para os desenvolvedores (FERNANDES, 2005).

A falta de um padrão estabelecido é um fator impeditivo para que uma empresa venha a investir em determinada tecnologia. O mercado tecnológico apresenta muitas tendências, sen-

do que boa parte das mesmas não sobrevive e é descontinuada. Nesse sentido, um erro a ser evitado, principalmente em ambientes de computação heterogênea: padrões proprietários. Faz-se necessário uma prévia análise da tecnologia de desenvolvimento, considerando principalmente a utilização de padrões abertos, que são mais adaptáveis nestes ambientes (BONIAT, 2005).

Neste contexto, podemos destacar que a característica da heterogeneidade nos sistemas distribuídos baseados em objetos, impõe a necessidade de especificações abertas, com interfaces padronizadas e públicas, levando ao desenvolvimento de *middlewares*.

Um *middleware* é uma camada de software, residente acima do sistema operacional e do substrato de comunicação, que oferece abstrações de alto nível, com objetivo de facilitar a programação distribuída. As abstrações oferecidas fornecem uma visão uniforme na utilização de recursos heterogêneos existentes nas camadas inferiores – sistema operacional, hardware e redes (BONIAT, 2005).

Os *middlewares* provêem um conjunto comum de serviços, que permitem que aplicações servidoras e clientes interajam através da rede. Possui uma importância fundamental para a programação distribuída, principalmente pelo fato de prover serviços padronizados.

Existem atualmente vários tipos de *middlewares*, mas a classificação que mais nos interessa é a dos orientados a objetos. Baseados fortemente em tecnologia e protocolos de objetos, esses produtos objetivam promover um ambiente transacional distribuído e heterogêneo, em que códigos transformados em objetos ou em componentes serão usados para compor transações. Também propõem oferecer, além da transparência de localização e de chamada de código heterogêneo

que a resposta vai ser enviada separadamente da requisição.

O padrão *Web services* fornece uma maneira aberta para integrar aplicações tanto dentro como fora das organizações. O uso de padrões abertos torna possível integrar componentes e aplicações de forma independente da tecnologia utilizada. Sua principal vantagem, em relação às demais tecnologias para sistemas distribuídos baseados em objetos, é a simplicidade e a utilização de padrões abertos, não proprietários (FERNANDES, 2005).

Antes do advento dos *Web services*, a integração das aplicações corporativas era muito difícil, quando se tratando das diferenças de linguagem de programação e *middleware* utilizados nas organizações. As chances de dois sistemas corporativos utilizarem a mesma linguagem de programação e/ou o mesmo *middleware* é quase que inexistente, pois não existe uma tecnologia melhor ou pior, e sim, na maioria das vezes, a preferida, ou com maior capacitação técnica das equipes na implementação. Essa “Guerra de Componentes” traduziu em muitas dores de cabeça na hora do esforço de integração que resultou em uma epidemia de soluções customizadas e integrações offline. Resumindo, interoperabilidade incômoda, dolorosa e difícil (BONIAT, 2005).

Web service é uma tecnologia para comunicação entre sistemas muito usada em aplicações B2B (Business to Business). A comunicação entre os serviços é padronizada, possibilitando a independência de plataforma e de linguagem de programação. Por exemplo, um sistema de reserva de passagens aéreas feito em Java e rodando em um servidor Linux pode acessar, com transparência, um serviço de reserva de hotel feito em .Net rodando em um servidor Microsoft (WEB..., 2005).

Os *Web Services* representam a evolução de alguns padrões e protocolos de larga utilização e empregados para criar a Internet como a conhecemos hoje. A padronização tem sido a chave para o sucesso e ampla aceitação da tecnologia. Sua infra-estrutura é construída sobre os padrões e tecnologias abertas, devidamente definidas e de ampla aceitação, como XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (*Web services* Description Language) e UDDI (Universal Description, Discovery and Integration). Sua implementação é possível e portátil sobre qualquer sistema operacional e linguagem de programação que suporte esses padrões.

Os *Web services* são divididos em cinco blocos: Publicação, Descrição, Formato da mensagem, Codificação e Transporte (BARBOSA, 2004).

- **Publicação:** esta camada é responsável por fazer a exposição dos serviços para a Internet, uma vez que a centralização dos serviços em um único local facilita a distribuição. Um exemplo destes concentradores é a UDDI (Universal Description Discovery and Integration). Um outro nome dado a este tipo de catálogo é páginas amarelas de serviços da Internet. Para que o *Web service* obtenha sucesso, é necessário publicá-lo no UDDI.
- **Descrição:** local aonde os *Web services* descrevem toda a sua documentação sobre caminho, serviços, parâmetros de entrada e saída. Esta descrição fica armazenada em um arquivo chamado WSDL, o qual está totalmente escrito XML utilizando os Schemas dos mesmos. Normalmente os desenvolvedores de *Web service*, aproveitam esta camada para fazer pequenos tutoriais de utilização dos seus serviços, uma vez que ao publicar um *Web*

cações servidoras e clientes interajam através da rede. Existem atualmente vários tipos de *middlewares*, dentre eles: DCOM, RMI, CORBA e WEB SERVICES.

A tecnologia DCOM objetiva suportar a comunicação entre objetos em diferentes computadores, definindo um modo padronizado para um módulo cliente e um módulo servidor se comunicarem através de uma interface específica. Ela foi idealizada visando a integração de ambientes heterogêneos, de forma a permitir a reutilização de ferramentas e objetos, diminuindo, assim, custos de novas aquisições e também para maior aproveitamento da produção da equipe de desenvolvimento já existente. Podemos considerar a tecnologia excelente para interação de sistemas sobre a plataforma Microsoft, sem interação com outros sistemas operacionais. Inexiste a necessidade de instalação de *drivers* nas máquinas clientes, uma vez que toda a comunicação entre cliente e servidor é realizada por meio de bibliotecas do sistema operacional. Mesmo que a tecnologia seja gratuita, ela só funciona sobre plataformas proprietárias que não o são.

Já o Java RMI foi a primeira arquitetura de objetos distribuídos suportada pela Máquina Virtual Java (JVM - *Java Virtual Machine*), através de seu *framework*. Foi projetada para permitir a invocação de métodos remotos de objetos em máquinas virtuais distintas, de maneira transparente. Embora o Java RMI permita a interoperabilidade entre aplicações em diferentes arquiteturas de hardware e sistemas operacionais, isso só é garantido com a utilização da JVM. Pelo fato de que só existem implementações da máquina virtual para linguagem Java, o desenvolvedor prende seus objetos a esta. Por outro lado, o desenvolvedor passa a ter toda a simplicidade e transparência da linguagem Java, programando em um elevado nível de abstração, encapsulando detalhes da distribuição (comunicação).

O CORBA é uma arquitetura de desenvolvimento baseada no modelo Cliente/Servidor, em que clientes solicitam chamadas de funções, localizadas em servidores remotos (ou locais). A localização desses, porém, não é indicada por meio de um *host/soquete* gerenciado pelo cliente, mas, sim, por meio de uma “referência de objeto”, usada para encaminhar corretamente os pedidos pelo “corretor de pedidos” (ORB – *Object Request Broker*), o qual provê um ambiente de comunicação que gerencia a solicitação de serviços e envio de mensagens. A adoção de CORBA tem tido grande sucesso em redes locais e corporativas onde as restrições de segurança são planejadas. Entretanto, na Internet, necessitamos de alterações específicas em *firewalls*, o que nem sempre é possível. Outro problema é que, embora CORBA seja uma especificação aberta, isso não garante totalmente a padronização.

Já os *Web services*, no sentido geral do termo, significam serviços oferecidos pela Internet, onde o uso de padrões abertos torna possível integrar componentes e aplicações de forma independente da tecnologia utilizada. A padronização tem sido a chave para o sucesso e aceitação instantânea da tecnologia. Sua infra-estrutura é construída sobre os padrões e tecnologias abertas, devidamente definidas e de ampla aceitação, como XML (*eXtensible Markup Language*), SOAP (*Simple Object Access Protocol*), WSDL (*Web service Description Language*) e UDDI (*Universal Description, Discovery and Integration*).

De acordo com Gunzer (2005), a linguagem de marcação extensível é um subconjunto da SGML (*Standard Generalized Markup Language*). Através dela, é possível descrever os objetos, seus atributos, métodos e parâmetros, assim como permitir que seus dados sejam interpretados pelas aplicações. O SOAP define um formato padrão, para codificação de dados, e uma estrutura simples, para expressar a semân-

dades específicas e que possam ser utilizados pela aplicação que está sendo criada. Analogamente, trata-se de um serviço similar ao prestado pelas páginas amarelas, exceto pelo fato de serem as mesmas, eletrônicas. Estas provêm uma estrutura de informações, na qual várias entidades de negócios registram-se elas mesmas e seus serviços respectivos, através de suas definições na linguagem WSDL, discriminadas por áreas de negócios como indústria, produto ou geograficamente, de forma a permitir uma busca otimizada. A especificação UDDI consiste de um esquema de quatro camadas dispostas hierarquicamente, as quais provêm um modelo base de informação para publicar, validar e invocar informações a respeito de um *Web service*. São elas: entidade de negócio (conjunto inicial de informações), serviço de negócio (relacionado a processos), informações de troca (endereço URL e servidor) e informações de concordância – para validar a especificação do serviço (GUNZER, 2005a).

Posta a conceituação destas três tecnologias, basicamente, o modelo de funcionamento segue uma linha que remete ao seguinte fluxo:

- Uma aplicação é desenvolvida, utilizando-se de qualquer tipo de linguagem de programação;
- Tal aplicação possui uma série de interfaces formalizadas, as quais são publicadas em WSDL;
- A entidade responsável pela aplicação registra a si mesma apropriadamente como sendo uma entidade de negócios e publica suas interfaces em WSDL em um ou mais registros UDDI, que vêm a ser diretórios de Servidores Web;
- Qualquer outra aplicação que necessita das funcionalidades daquela desenvolvida pes-

quisa os registros UDDI e localiza os serviços correspondentes.

Dessa forma, aplicações distribuídas podem ser criadas, estando localizadas em diferentes servidores e mesmo em empresas diversas. Sistemáticamente, inseridos em uma mesma empresa temos ambientes diferentes, acarretando com isso dificuldades de resolução quanto ao compartilhamento de informações e integração das aplicações. Como visto anteriormente, a solução para este caos tecnológico teve seu início quando do surgimento da Internet, através da padronização em diversos segmentos: TCP/IP como protocolo da Internet e redes internas, HTML, como formato padrão para a publicação de informações e HTTP como protocolo de transporte. Mais recentemente surgiu o XML para atender, ainda que parcialmente, a troca de informações entre as empresas que utilizam formatos de dados proprietários e incompatíveis. A aceitação deste padrão não foi total, pois reside ainda o problema de como fazer com que uma aplicação possa utilizar-se de funcionalidades de outras aplicações sem a necessidade de se reescrever a lógica das mesmas (FERNANDES, 2005).

Então, Matsumoto (2005) cita os *Web services*, que fornecem ao ambiente corporativo a padronização de meios de acesso, possibilitando às empresas disponibilizar suas aplicações através de serviços *Web services*. Desta forma, eles possibilitam que aplicações desenvolvidas em linguagens diferentes se comuniquem através de uma especificação comum.

Na definição do contexto, o autor diz que *Web service* é um termo dado a um serviço disponibilizado através de uma aplicação servidora para clientes, via World Wide Web. Ele funciona com um cliente enviando uma requisição para os serviços (*Web services*) através de uma

