3
Serviços na Web (*Web services*)

3.1.
Visão Geral

Com base na definição do Word Wide Web Consortium (W3C), *web services* são aplicações autocontidas, que possuem interface baseadas em XML e que descrevem uma coleção de operações acessíveis através de rede, independentemente da tecnologia usada na implementação do serviço [W3C, 2004].

Um *web service* pode ser entendido como um componente que possui suas funcionalidades acessíveis pela rede através de mensagens baseadas em XML. A disponibilização das operações e a descrição do serviço também ocorrem através do padrão XML. O arquivo descritor do serviço possui todas as informações necessárias para que outros componentes possam interagir com o serviço, incluindo o formato das mensagens (para as chamadas aos métodos do serviço), protocolos de comunicação e as formas de localização do serviço. Um dos maiores benefícios dessa interface é a abstração dos detalhes de implementação do serviço, permitindo que seja acessado independente da plataforma de hardware ou software na qual foi implementado. Como as mensagens trocadas para a comunicação são baseadas no padrão XML, também temos a flexibilidade com relação à linguagem de programação tanto na implementação do serviço quanto no componente que acessará o *web service*. Estas características permitem e motivam a implementação de aplicações Web baseadas em *web services* por torná-las fracamente acopladas com as outras partes do código da aplicação. Com isso, as aplicações adquirem uma arquitetura componentizada e tornam-se flexíveis com relação às várias plataformas disponíveis no mercado. Um *web service* geralmente é implementado para disponibilizar uma determinada funcionalidade autocontida visando a reusabilidade do *web service* e a interoperabilidade com outros sistemas.

Neste contexto, imagina-se a possibilidade de um sistema utilizar um ou mais *web services* para realizar uma determinada transação, criando um esquema de composição de *web services*. Com a adoção dessa tecnologia, nota-se a tendência de criação de *web services* "inteligentes" com o uso de agentes de software, flexíveis e dinâmicos, que podem se adaptar ao contexto de cada requisição e produzir resultados dinâmicos para cada situação.

3.2. Arquitetura de Web Services

Uma definição técnica de *web services* poderia ser como um serviço disponibilizado na Internet, descrito via WSDL, registrado via UDDI, acessado utilizando SOAP e com os dados transmitidos sendo representados em XML. A seguir, encontra-se uma breve explicação de algumas tecnologias citadas na definição anterior:

SOAP (*Simple Object Access Protocol*) é um protocolo para troca de informações em ambiente distribuído. É baseado em definições XML e utilizado para acessar *web services*. Esse protocolo encapsula as chamadas e retornos aos métodos dos *web services*, sendo utilizado, principalmente, sobre HTTP.

WSDL (*Web Services Description Language*) é a linguagem de descrição de *web services* baseada em XML. Ela permite, através da definição de um vocabulário em XML, a possibilidade de descrever serviços e a troca de mensagens. Mais especificamente é responsável por prover as informações necessárias para a invocação do *web service*, como sua localização, operações disponíveis e suas assinaturas.

UDDI (*Universal Description, Discovery and Integration*) é uma das tecnologias que possibilitam o uso de *web services*. Uma implementação de UDDI corresponde a um *Web Service registry*, que provê um mecanismo para busca e publicação *web services*. Um *UDDI registry* contém informações categorizadas sobre os serviços e as funcionalidades que eles oferecem, e permite a associação desses serviços com suas informações técnicas, geralmente definidas usando WSDL. Como dito anteriormente, o arquivo de descrição em WSDL descreve as funcionalidades do *web service*, a forma de comunicação e sua localização. Devido ao modo de acesso, um *UDDI registry* também pode ser entendido como

um *web service*. A especificação UDDI define uma API baseada em mensagens SOAP, com uma descrição em WSDL do próprio *web service* do servidor de registro. A maioria dos servidores de registro UDDI também provê uma interface de navegação por *browser*.

A arquitetura de *web services* se baseia na interação de três entidades: provedor do serviço (*service provider*), cliente do serviço (*service requestor*) e servidor de registro (*service registry*). De uma forma geral, as interações são para publicação, busca e execução de operações. A figura 7 ilustra estas operações, os componentes envolvidos e suas interações.

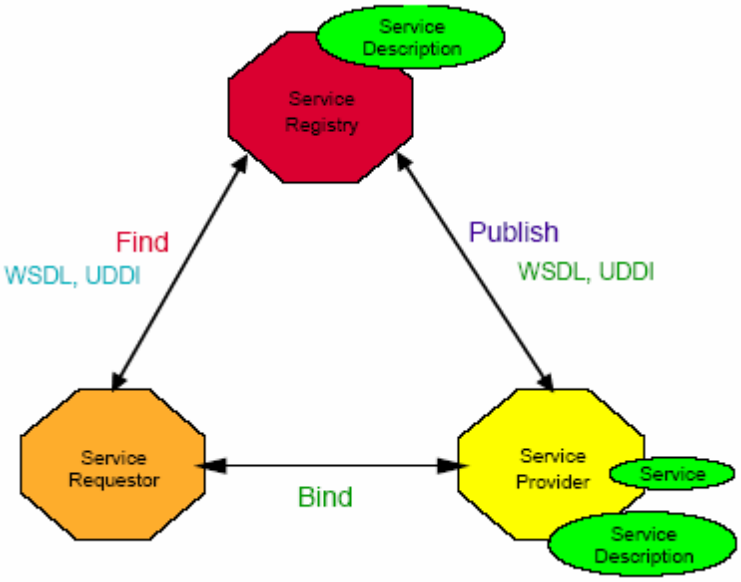

Figura 1 – Arquitetura de web services, operações e interação das entidades.

O provedor do serviço (*service provider*) representa a plataforma que hospeda o *web service* permitindo que os clientes acessem o serviço. O cliente do serviço (*service requestor*) é a aplicação que está procurando, invocando ou iniciando uma interação com o *web service*. O cliente do serviço pode ser uma pessoa acessando através de um *browser* ou uma aplicação realizando uma invocação aos métodos descritos na interface do *web service*. O *service registry* representa os servidores de registro e busca de *web services* baseados em arquivos de descrição de serviços que foram publicados pelos *service providers*. Os clientes (*service requestor*) buscam por serviços nos servidores de registro e recuperam informações referentes a interface de comunicação para os *web services* durante a fase de desenvolvimento ou durante a execução do cliente, denominados *static binding* e *dinamic binding*, respectivamente.

3.2.1.
Plataformas para Web Services

A disseminação no uso de *web services* nos últimos anos incentivou o mercado a oferecer uma grande variedade de ferramentas e aplicações para prover suporte a essa tecnologia. Atualmente, as principais plataformas para *web services* são: Sun Microsystems, IBM, BEA, Apache, Systinet e Microsoft. A seguir, encontra-se um resumo dos principais fornecedores citados e seus produtos. O objetivo é apenas citar as tecnologias para *web services* disponíveis durante a realização desse trabalho de pesquisa.

A IBM é um fornecedor altamente diversificado de tecnologias e ferramentas para *web services* e servidores de aplicação. O IBM WebSphere Software Developer Kit for Web Services (WSDK V5.0.1) é um conjunto de ferramentas para a criação, busca, invocação e testes para *web services*. O WSDK versão 5.0.1 encontra-se em conformidade com as últimas versões das especificações para *web services* incluindo WS-Security, SOAP, WSDL e UDDI. Entre as principais ferramentas de desenvolvimento, podem ser citadas o *WebSphere Studio Application Developer* (IDE), *Web Services Gateway*, *Web Services Invocation Framework* e o *UDDI Explorer*.

A Microsoft disponibilizou ferramentas para a tecnologia de *web services* no seu pacote de produtos Visual Studio .NET e colabora para a publicação das principais especificações de *web services*.

O pacote de produtos da BEA para *web services* tem sua arquitetura baseada em J2EE e é composto por ferramentas como IDE (ambiente de desenvolvimento), UDDI registry e o UDDI browser. Tais produtos já são compatíveis com a especificação SOAP 1.2. A BEA tem contribuído para a especificação Web Services Choreography Interface (WSCI) em parceria com a Sun Microsystems, e para as especificações WS-Coordination, WS-Transaction e Business Process Execution Language for Web Services (BPEL4WS). O servidor de aplicação da BEA é o Weblogic Server 7.0.

A Systinet é uma empresa que difunde a tecnologia de *web services* e oferece uma plataforma independente que já se encontra em conformidade com a última especificação SOAP 1.2 proposta pelo World Wide Web Consortium

(W3C). A Systinet disponibiliza um completo conjunto de ferramentas para implementação, implantação, segurança e gerenciamento de *web services*.

O Java Web Services Developer Pack (Java WSDP) é um conjunto de ferramentas integradas e gratuitas que permite desenvolvedores Java implementarem e testarem aplicações web com XML e *web services*. O Java WSDP engloba as APIs de Java para XML, Java Architecture for XML Binding (JAXB), JavaServer Faces, Web Services Interoperability Sample Application, Web Services Security, JavaServer Pages Standard Tag Library (JSTL) e Java WSDP Registry Server. As ferramentas Ant Build Tool e Apache Tomcat container também fazem parte deste pacote distribuído pela Sun. O servidor de aplicação da Sun é o Sun ONE™ Application Server 7.0

Apache Axis é um projeto open source para um servidor e cliente SOAP. Axis é considerado basicamente um SOAP engine – um *framework* para a construção de processadores de mensagens SOAP. Atualmente está implementado apenas na linguagem Java, mas uma versão para cliente C++ está em andamento.

3.3. Reusabilidade e Interoperabilidade de Web Services

Web services são a mais recente evolução no desenvolvimento de aplicações para internet e a importância do padrão XML é evidente quando consideramos as vantagens de fácil integração, interoperabilidade entre sistemas e uma comunicação independente de especificações proprietárias de um fornecedor específico.

A conseqüência esperada devido ao surgimento de *web services* é o desenvolvimento de aplicações que podem ser conceituadas como um conjunto de serviços passíveis de serem publicados, encontrados por ferramentas de busca, combinados e consumidos para gerar os resultados esperados. Tais características são razões para o crescente interesse em *web services*, que permite a possibilidade de modelar e implementar uma Arquitetura Orientada a Serviços (SOA - *Service-Oriented Architecture*). Quando se utiliza *web services* para a construção de tal arquitetura, as soluções consistem de coleções de serviços autônomos, identificados por URL, com interfaces documentadas através de WSDL e

processando mensagens XML. SOA é uma solução complementar com relação a abordagens orientadas a objetos (OO) e procedurais.

Um cenário idealizado com base nas afirmações acima é a criação de aplicações que podem ser reestruturadas de forma dinâmica através de *web services* para necessidades específicas. Neste sentido, serviços podem ser considerados “objetos” funcionais independentes que podem interagir com outros serviços.

Essa última observação induz à reflexão e comparação com base nas definições de *learning objects* citadas no capítulo anterior. Essa reflexão será muito valiosa para o entendimento do próximo capítulo, onde é explicado com mais detalhe essa interseção entre as duas tecnologias.