UNIVERSIDADE DO VALE DO RIO DOS SINOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM
COMPUTAÇÃO APLICADA - PIPCA

GlueScript: Uma Linguagem Específica de Domínio para Composição de *Web Services*

por

ROSELI PERSSON HANSEN

Dissertação submetida a avaliação
como requisito parcial para a obtenção do grau de
Mestre em Computação Aplicada

Prof. Dr. Sérgio Crespo Coelho da Silva Pinto
Orientador

São Leopoldo, Fevereiro de 2003

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Hansen, Roseli Persson

GlueScript: Uma Linguagem Específica de Domínio para Composição de *Web Services* / por Roseli Persson Hansen. — São Leopoldo: Centro de Ciências Exatas e Tecnológicas da UNISINOS, 2003.

89 f.: il.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos. Centro de Ciências Exatas e Tecnológicas Programa Interdisciplinar de Pós-Graduação em Computação Aplicada - PIPCA, São Leopoldo, BR–RS, 2003. Orientador: Pinto, Sérgio Crespo Coelho da Silva.

1. *Web Services*. 2. Metadados Educacionais. 3. Educação a Distância. 4. Linguagens de Composição. 5. Linguagem Específica de Domínio. I. Pinto, Sérgio Crespo Coelho da Silva. II. Título.

"Dedico esta dissertação ao meu marido Cristiano,
aos meus pais Rui e Rita,
aos meus irmãos Romir e Ronado,
a minha cunhada Amaraí e
aos meus dois sobrinhos Rodrigo e Robson."

Agradecimentos

Esta dissertação não teria sido realizada sem o apoio e a ajuda de algumas pessoas que foram indispensáveis. A todos eu agradeço do fundo do meu coração.

Aos meus familiares que sempre estiveram presentes compartilhando das alegrias e me incentivando nas horas mais difíceis. Especialmente ao meu pai Rui Persson e a minha mãe Rita Persson, que nos momentos de tristeza foram maravilhosos tendo sempre um conselho e uma palavra de apoio para que eu concluísse a minha dissertação.

Ao meu marido Cristiano Roberto Hansen que esteve sempre ao meu lado dando todo o suporte necessário para o desenvolvimento do meu trabalho. Sempre além do incentivo, demonstrou atenção e carinho, os quais não esquecerei.

Ao professor Sérgio Crespo C. da S. Pinto por ser meu orientador. Apesar das dificuldades, incetivou-me no desenvolvimento desta dissertação auxiliando com comentários, críticas e importantes contribuições para que eu chegasse no final deste trabalho.

Às professoras Lúcia Maria Martins Giraffa e Renata Vieira por participarem da minha banca.

À minha querida amiga Jacira Dopke que fez a revisão da minha dissertação.

Aos colegas do mestrado com os quais foi possível compartilhar os bons e os maus momentos no decorrer destes dois anos.

À CAPES pelo apoio financeiro necessário à realização desta dissertação.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ADL	Advanced Distributed Learning
ALI	ARIADNE Learner Interface
AMI	ARIADNE Management Interface
API	Aplpication Program Interface
ARIADNE	Alliance of Remote Instructional Authoring and Distribution Networks for Europe
CLR	Common Language Runtime
CORBA	Common Object Request Broker Arquitecture
DAD	Document Access Definition
DADX	Document Access Definition Extension
DCMI	Dublin Core Metadata Inciative
DCOM	Distributed Component Object Model
DOM	Document Object Model
DSL	Domain Specific Language
DTD	Document Type Definition
ebXML	e-business XML
EdNA	Education Network Australia
EJB	Enterprise Java Beans
FTP	File Transfer Protocol
GEM	Gateway to Educational Materials
GEMSTONES	Gestalt Extensions to Metadata Standards for ON-line Educatinal Systems
GESTALT	Getting Educational Systems talking Across Leading-Edge Tecnologies
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IE	Informática na Educação

IEEE	Institute of Electrical and Eletronics Engineers
IIOP	Internet Inter-Orb Protocol
IMS	Instructional Management Systems
JAXM	Java API for XML Message
JAXP	Java API for XML Parsing
JAXR	Java API for XML Registry
JAX-RPC	Java API for XML Remote Procedure Call
JDBC	Java Database Connectivity
J2EE	Java2 Enterprise Edition
JRE	Java Runtime Environment
JSP	Java Server Pages
JWSDP	Java Web Services Developer Pack
KPS	knowledge Pool System
LMS	Learning Management Systems
LOM	Learning Object Metadata
LTSC	Learning Technologies Standards Commitee
MIME	Multi Purpose Mail Extensions
NDSL	National SMETE Digital Library
PAD	Personal Digital Assistent
QTI	Question and Test Interoperability
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SAX	Simple API XML
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
UDDI	Universal Description, Discovery, and Integration
VASP	Value Added Service Provider
WML	Wireless Markup Language
WSDL	Web Services Description Language
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSTL	Stylesheet Language Transformations

Resumo

Esta dissertação apresenta uma proposta de solução para a criação de ambientes para Web. Apresenta algumas tecnologias como *Web Services* e metadados educacionais. A partir destas tecnologias foi criada uma proposta de arquitetura genérica a fim de integrar as mesmas através de uma linguagem específica de domínio (DSL) chamada GlueScript. Esta linguagem permite a composição de *Web Services* e recursos educacionais para a construção de ambientes para educação a distância.

Foi realizado um estudo de caso, de forma a verificar a usabilidade da linguagem GlueScript.

Abstract

This master dissertation presents a proposal to creation environments for Web application. It introduces some technologies as Web Services and educational metadata. Starting from these technologies it was created a generic architecture to integrate them through a domain specific language (DSL) called GlueScript. This language allows to composit of Web Services and educational resources for the construction of distance education applicattion.

A case study was built, in way to test the GlueScript language.

Capítulo 1

Introdução

Resumo

Este capítulo apresenta uma introdução possibilitando uma visão geral do trabalho a ser realizado, destacando-se a motivação para a realização do mesmo, bem como o objetivo a ser alcançado.

1.1 Introdução

A educação a distância consiste no ensino por meio de mídia impressa ou eletrônica para pessoas engajadas em um processo de aprendizado em tempo e local diferente do(s) instrutor(es) e dos outros aprendizes.

A educação baseada na Web utiliza a WWW como meio de publicação do material didático, aplicação de tutoriais, aplicação de provas e testes e comunicação com estudantes. Também compreende o processo de uso da Web como veículo de comunicação para a apresentação de aulas a distância. As tecnologias de educação/treinamento, baseada na Web, estão em pleno uso, utilizando uma série de ferramentas que auxiliam os processos de cooperação, coordenação e comunicação ([56, 50]).

Com o crescimento da Internet, surgem novas tecnologias para a disponibilização de serviços. Na Internet, existem no momento cerca de 605 milhões de usuários e 4,3 milhões de Web *Sites* ([63]). Assim, surge a necessidade de tornar estes serviços interoperáveis e reutilizáveis, o que pode ser feito através da utilização de *Web Services* ou metadados educacionais, por exemplo. Deste modo, evolui também a forma de ensino já que os ambientes educacionais precisam adaptar-se a estas novas tecnologias.

A partir da necessidade de tornar os serviços interoperáveis e reutilizáveis, certas corporações estão desenvolvendo uma arquitetura que permite reuso de serviços. Algumas empresas, como Sun, IBM, HP e Microsoft, estão desenvolvendo plataformas para criação e acesso a *Web Services*. Um *Web Services* é um componente de software independente de implementação e plataforma. Pode ser descrito utilizando uma linguagem de descrição de serviços, publicado em um registro e descoberto através de um mecanismo padrão. Pode também ser invocado a partir de uma *Application Program Interface* (API), através da rede e ser composto juntamente com outros serviços.

A tecnologia de *Web Services* permite que serviços sejam acessados e reutilizados através do uso de tecnologias padrões como: *Web Services Description Language* (WSDL), *Simple Object Access Protocol* (SOAP) e *Universal Description, Discovery and Integration* (UDDI). Estas tecnologias são baseadas em *eXtensible Markup Language* (XML) e permitem invocar, ou reutilizar, um serviço sem a necessidade de conhecer a plataforma ou a linguagem de programação usada na sua construção.

Por outro lado, os sistemas educacionais estão direcionando seus esforços na tentativa de obter maior interoperabilidade dos recursos entre diferentes sistemas. Para tornar esta interoperabilidade possível, torna-se necessário a padronização dos metadados ([56]).

Os recursos educacionais disponibilizados na Web, com o propósito de reuso e interoperabilidade são descritos através de metadados, como por exemplo o *Learning Object Metadata* (LOM) e o metadado *Dublin Core*. Os metadados são descrições feitas no padrão XML, o que possibilita a integração com a tecnologia de Web Services.

1.2 Motivação e Contexto do Trabalho

Com a evolução das tecnologias para Web, permitiu-se a expansão das alternativas para suporte ao processo de ensino e aprendizagem, principalmente a educação a distância a qual tomou um novo rumo fazendo uso destas tecnologias. Os ambientes educacionais para Web necessitam se adaptar às tecnologias emergentes, de forma a fazer o melhor uso e alcançar o máximo de benefícios possíveis.

Logo, existe na comunidade de Informática na Educação (IE) o objetivo de facilitar o desenvolvimento de cursos Web, ou outras aplicações voltadas para educação a distância pode trazer inúmeros benefícios assim como reuso, por exemplo. A reutilização de serviços e recursos para a construção de novas aplicações, reduz de forma acentuada os custos de implementação. Dentro deste contexto o trabalho foi desenvolvido.

1.3 Problema

As arquiteturas educacionais existentes têm uma definição bastante precisa quanto à descrição de recursos educacionais, que utilizam LOM-XML. A arquitetura de *Web Services,* em geral, apresenta uma estrutura bem definida para permitir busca, acesso e principalmente reuso de serviços na Web.

Entretanto, o fator relevante do desenvolvimento deste trabalho está em permitir utilizar a tecnologia de *Web Services* como mecanismo de busca e acesso a serviços e recursos educacionais. Os recursos educacionais são muito bem especificados através de metadados, considerando os aspectos pedagógicos, o que não é possível através da descrição padrão utilizada na tecnologia *Web Services*.

O desenvolvimento de aplicações Web destinadas a educação a distância nem sempre é uma tarefa simples. Integrar duas tecnologias que evoluem em paralelo, através de uma linguagem de composição, traz menor custo e maior rapidez no desenvolvimento destas aplicações. Isto torna-se possível através da reutilização de serviços e recursos já existentes.

1.4 Questão de Pesquisa

Com o foco no desenvolvimento de aplicações Web voltadas para a educação, surge a possibilidade de construir ambientes na Web, de forma rápida e eficiente, a partir da reutilização de serviços e recursos já existentes.

Logo, a questão de pesquisa que norteia este trabalho é integrar as tecnologias de *Web Services* e de metadados educacionais que utilizam LOM-XML. Permitir que através desta integração seja possível construir ambientes que possam ser utilizados para educação a distância. Isso torna-se possível no momento em que será desenvolvida uma linguagem específica de domínio para composição de *Web Services* e recursos educacionais.

1.5 Objetivo

Este trabalho apresenta uma linguagem de composição de serviços e recursos educacionais tornando possível reutilizar *Web Services* e recursos educacionais disponíveis na Web, a partir de uma proposta de arquitetura a qual visa permitir que recursos disponibilizados por sistemas educacionais possam ser reutilizados juntamente com serviços acessados através da arquitetura de *Web Services*.

Isto permite a construção de aplicações para educação a distância com alto grau de reuso. A linguagem de composição objetiva permitir integrar as duas tecnologias de forma que *Web Services* podem ser compostos juntamente com recursos educacionais.

1.6 Organização do Trabalho

Este volume está organizado em sete capítulos, sendo que o Capítulo 2 apresenta uma visão geral de *Web Services*, abordando as principais iniciativas, focando suas arquiteturas e propostas.

O Capítulo 3 apresenta um estudo sobre metadados educacionais, como estes são modelados e quais as principais arquiteturas. Para isto são vistos metadados como LOM, Dublin Core e algumas extensões destes metadados, apresentadas pelas principais iniciativas de padronização de metadados.

No capítulo 4 é apresentada uma arquitetura genérica para composição de serviços educacionais. Esta arquitetura possibilita a composição de serviços com enfoque educacional. É composta pela *Domain Specific Language* (DSL) *GlueScript*, apresentada no Capítulo 5, a qual pemite a composição de serviços e recursos educacionais utilizando-se a tecnologia de *Web Services* e descrição de recursos educacionais através do metadado LOM.

O Capítulo 6 apresenta um estudo de caso, sendo que para validar a arquitetura proposta, a linguagem foi utilizada por desenvolvedores Web para construir aplicações direcionadas à educação. Também uma avaliação do uso da linguagem, descrevendo como esta foi utilizada e os resultados obtidos.

No Capítulo 7 são apresentadas as conclusões da realização do trabalho, as contribuições e os trabalhos futuros.

Capítulo 2

Visão Geral de *Web Services*

Resumo

Web Services apresentam uma estrutura arquitetural que permite a comunicação entre aplicações. O uso de tecnologias baseadas em XML para construção de *Web Services,* permite a utilização de serviços sem que haja necessidade de saber qual plataforma ou linguagem de programação foi utilizada na sua contrução. Diferentes arquiteturas e plataformas estão sendo propostas. Este capítulo procura apresentar uma visão geral das principais abordagens sobre arquiteturas de *Web Services* e estabelecer quais são os pontos em comum entre elas.

2.1 Introdução

Os *Web Services* apresentam uma estrutura arquitetural que permite a comunicação entre aplicações. Um serviço pode ser invocado remotamente, ou ser utilizado para compor um novo serviço juntamente com outros. Além da arquitetura básica de *Web Services*, estarão sendo apresentadas algumas tecnologias padrões para construção de *Web Services* tais como: WSDL [20], SOAP [12] e UDDI [24]. Estas tecnologias são baseadas em XML [13], e permitem invocar ou reutilizar um serviço sem a necessidade de conhecer a plataforma ou linguagem de programação usada na sua construção. Aborda-se, ainda, as principais arquiteturas de *Web Services* propostas por: Microsoft, HP, Sun e IBM.

Apresenta uma breve conceituação de *Web Services*, bem como os componentes de uma estrutura básica (Registro de Serviços, Solicitante de Serviços e Provedor de Serviços). Este capítulo descreve as principais tecnologias padrões utilizadas em *Web Services*. Também são descritas as principais abordagens sobre as arquiteturas e algumas plataformas de desenvolvimento de *Web Services* que estão sendo disponibilizadas.

2.2 O que são *Web Services*?

Os *Web Services* são aplicações modulares que podem ser descritas, publicadas e invocadas sobre uma rede, geralmente a Web. Ou seja, é uma interface que descreve uma coleção de operações que são acessíveis pela rede através de mensagens em

formato XML padronizadas. Os *Web Services* permitem uma integração de serviços de maneira rápida e eficiente ([48, 19]).

Um *Web Service* é um componente de software independente de implementação e plataforma. Pode ser descrito utilizando-se uma linguagem de descrição de serviços, publicado em um registro e descoberto através de um mecanismo de busca padrão. Por conseguinte, também ser invocado a partir de uma *Application Program Interface* (API) através da rede, e ser composto com outros serviços.

Os *Web Services* combinam os melhores aspectos do desenvolvimento baseado em componentes na Web. Assim como componentes de *software*, os *Web Services* representam uma funcionalidade *black-box* que pode ser reutilizada sem a preocupação com a linguagem utilizada no seu desenvolvimento ([30]).

O termo *Web Services* descreve funcionalidades específicas do "negócio" exposto usualmente através de conexões internet, com o propósito de fornecer um caminho para a utilização deste serviço.

A exposição dos serviços envolve:

- Identificar ou definir funções de valor no negócio ou processos.
- Definir uma interface baseada no serviço para os processos.
- Descrever estas interfaces em um formato baseado na Web.

Para tornar um serviço disponível sobre a Web normalmente é necessário:

- Publicar a interface do serviço para que este possa ser encontrado e utilizado.
- Aceitar requisições e enviar respostas usando protocolo padrão e mensagens em formato XML.
- Fazer uma ligação entre requisições externas e implementações das funções dos negócios.

Utilizando um documento criado na forma de uma mensagem, um programa envia uma requisição para um *Web Service* através da rede e opcionalmente recebe uma resposta, também na forma de um documento XML. O uso de XML torna-se importante por disponibilizar um mecanismo estensível para descrever as mensagens e os seus conteúdos. Assim sendo, padrões de serviços Web definem o formato da mensagem, bem como mecanismos para publicar e descobrir interfaces de *Web Services*.

Os *Web Services* não são acessados via protocolos específicos de modelos de objetos: *Distributed Component Object Model* (DCOM), *Remote Method Invocation* (RMI) ou *Internet Inter-ORB Protocol* (IIOP), mas são acessados por protocolos de transporte como *Hiper Text Transfer Protocol* (HTTP), *File Transfer Protocol* (FTP) e *Simple Mail Transfer Protocol* (SMTP).

2.2.1 Arquitetura de *Web Services*

Em uma arquitetura *Web Service* a descrição de um serviço cobre todos os detalhes necessários para que haja interação com um serviço, incluindo o formato das mensagens, os protocolos de transporte e a localização.

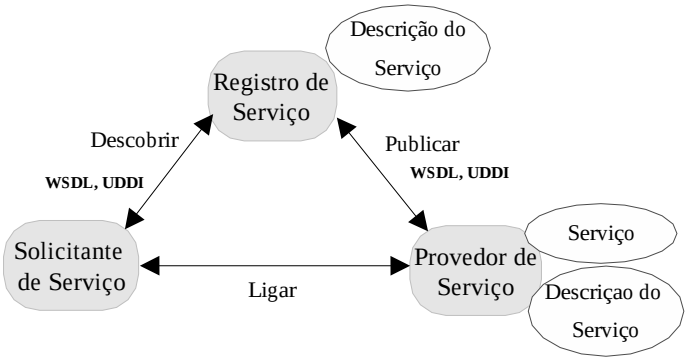

FIGURA 2.1 – *Web Services* papéis, operações e artefatos ([48])

A interface esconde os detalhes de implementação do serviço. Isto permite que as aplicações baseadas em *Web Services* sejam orientadas a componentes, e desta forma com mais potencial de reuso.

A arquitetura dos *Web Services,* apresentada na Figura 2.1, está baseada nas interações de três papéis:

- **Provedor de Serviço:** da perspectiva do negócio este é o dono do serviço. Da perspectiva arquitetural esta é a plataforma acessada na solicitação do serviço. É a entidade que cria o *Web Service* sendo responsável por fazer sua descrição em algum formato padrão e publicar os detalhes em um registro central.

- **Solicitante de Serviço:** é uma aplicação que invoca ou inicializa uma interação com um serviço. Pode ser um *browser* ou um programa sem interface com o usuário como por exemplo outro *Web Service.* Através da descrição de um serviço é possível encontrar e invocar *Web Services.*

- **Registro de Serviço:** é o local onde os provedores de serviços publicam suas descrições dos serviços. Os **Solicitantes de Serviços** procuram-nos e obtêm as informações de ligação da descrição do serviço, durante a fase de desenvolvimento (ligação estática) ou em tempo de execução (ligação dinâmica).

Como mencionado anteriormente, existe a **Descrição do Serviço** que contém os detalhes da interface e de implementação do serviço, incluindo a estrutura de dados, operações e informações de ligação na rede. Também contém dados para facilitar a sua localização pelo **Solicitante de Serviço**. O **Serviço** é o *software* disponibilizado na rede pelo **Provedor de Serviço**.

Além disso, há algumas operações nesta arquitetura que são:

- **Ligar:** quando um serviço necessita ser utilizado, a operação ligar invoca e inicializa a interação em tempo de execução, usando as informações de ligação da descrição do serviço.

- **Publicar:** para que um serviço possa ser acessado, este deve ser publicado no **Registro de Serviço**. O **Provedor de Serviço** contata o **Registro de Serviço** e publica um serviço.

- **Descobrir:** um **Solicitante de Serviço** encontra uma descrição do serviço ou consulta o **Registro de Serviço** para o tipo de serviço requerido. Um **Solicitante de Serviço** pode encontrar uma descrição da interface do serviço para o desenvolvimento de *software* na fase de *design* ou em tempo de execução. Desta forma, são encontradas as informações sobre ligações e localizações necessárias para invocar um serviço.

2.2.2 Camadas Conceituais de *Web Services*

A Figura 2.2 apresenta as camadas conceituais de *Web Services* descritas a seguir ([48], [37]):

- **Camada de rede:** é a camada base que representa protocolos como: HTTP, FTP e e-mail. Esta camada é utilizada de acordo com os requisitos da aplicação: segurança, disponibilidade, performance e confiabilidade.
- **Mensagem baseada em XML:** esta camada usa como padrão de tecnologia o protocolo SOAP para troca de informações em um ambiente distribuído.
- **Descrição do Serviço:** a descrição do serviço é feita utilizando-se WSDL, que define uma interface e mecanismos de interação dos serviços, além de descrições adicionais como contexto, qualidade do serviço e o relacionamento de serviço para serviço.
- **Publicação e Descoberta do Serviço:** estas duas camadas utilizam o registro UDDI para fazer a descoberta e a publicação de informações sobre *Web Services*.

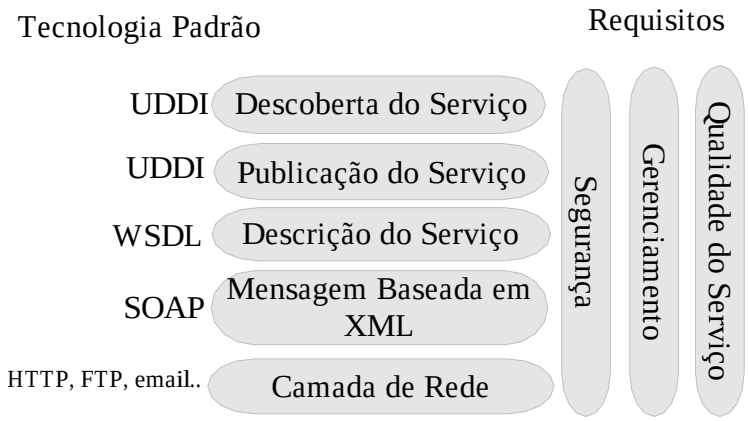

FIGURA 2.2 – Camadas conceituais de *Web Services* ([48])

2.3 Tecnologias Padrões Utilizadas nas Arquiteturas *Web Services*

Anteriormente foram mencionadas três tecnologias padrões para estruturação de *Web Services*. Dentre estas estão a WSDL, SOAP e UDDI abordadas de forma mais completa nas subseções que seguem.

2.3.1 *Web Services Description Language* (WSDL)

A WSDL é uma linguagem em formato XML, como apresentado na Figura 2.3, a fim de descrever serviços que podem ser definidos como um conjunto de operações acessíveis através de mensagens ([20]).

```
<?xml version="1.0" encoding="UTF- 8"?>

<definitions name="PerpetualCalendar" targetNamespace="urn:PerpetualCalendar/wsdl" xmlns:tns="urn:PerpetualCalendar/wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
 <types/>
 <message name="PerpetualCalendarServantInterface_getDayofWeek">
   <part name="year" type="xsd:string"/>
   <part name="month" type="xsd:string"/>
   <part name="day" type="xsd:string"/></message>
 <message name="PerpetualCalendarServantInterface_getDayofWeekResponse">
   <part name="result" type="xsd:string"/></message>
 <portType name="PerpetualCalendarServantInterface">
   <operation name="getDayofWeek" parameterOrder="year month day">
     <input message="tns:PerpetualCalendarServantInterface_getDayofWeek"/>
     <output message="tns:PerpetualCalendarServantInterface_getDayofWeekResponse"/></operation></portType>
 <binding name="PerpetualCalendarServantInterfaceBinding" type="tns:PerpetualCalendarServantInterface">
   <operation name="getDayofWeek">
     <input>
       <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="urn:PerpetualCalendar/wsdl"/></input>
     <output>
       <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="urn:PerpetualCalendar/wsdl"/></output>
     <soap:operation soapAction=""/></operation>
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/></binding>
 <service name="PerpetualCalendar">
   <port name="PerpetualCalendarServantInterfacePort" binding="tns:PerpetualCalendarServantInterfaceBinding">
     <soap:address location="http://localhost:8000/PerpetualCalendar/PerpetualCalendar"/></port></service></definitions>
```

FIGURA 2.3 – Exemplo de Documento WSDL

Define interfaces e mecanismos de interação dos serviços, protocolos de ligações e detalhes dos serviços da rede. Apresenta descrições adicionais como contexto, qualidade do serviço e o relacionamento de serviço para serviço. A WSDL permite que um objeto possa ser descrito de forma totalmente transparente de sua implementação.

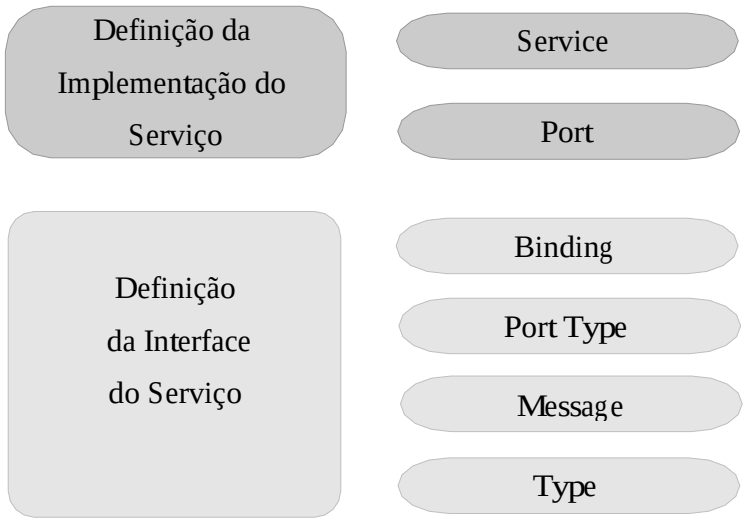

FIGURA 2.4 – Camada da descrição de serviços ([48])

O uso da WSDL permite a divisão da descrição dos serviços básicos em duas partes como mostra a Figura 2.4.

Esta definição básica do serviço é dividida em interface e implementação do serviço, permitindo que estas partes possam ser reutilizadas separadamente. A seguir, uma breve descrição das camadas da Figura 2.4 ([48, 20]).

Camada de Definição da Interface do Serviço

Uma interface de serviço contém a definição de serviço WSDL. Esta definição permite que uma interface possa ser utilizada, instanciada e referenciada por múltiplas definições de implementação de serviços ([14]).

O WSDL:*binding* descreve o protocolo, o formato dos dados, a segurança e outros atributos para uma interface de serviço particular (WSDL:*portType*). No WSDL:*portType* os elementos das operações do *Web Service* são definidos.

O WSDL:*message* é usado para definir os parâmetros de *input* e *output* de dados de uma operação.

O WSDL:*type* define o uso de tipos de dados complexos dentro de uma mensagem.

Um arquivo de interface descreve o *Web Service*, incluindo os métodos que são chamados, os parâmetros que são passados e a codificação utilizada.

Camada de Definição da Implementação do Serviço

É um documento WSDL que descreve como uma interface de serviço é implementada por um provedor de serviço. Um serviço Web é modelado como um elemento WSDL contendo uma coleção de WSDL: *port elements* (porta associada a localização de um serviço) com um elemento WSDL: *binding* de uma definição de interface.

Um arquivo de implementação descreve onde o *Web Service* está instalado e como este é acessado.

Além das definições de interface e implementação de serviço, a WSDL especifica extensões para ligação com protocolos e formatos de mensagem, como SOAP, HTTP GET/POST e MIME ([20]).

2.3.2 *Simple Object Access Protocol* (SOAP)

SOAP é um protocolo para troca de informações em um ambiente descentralizado e distribuído, permitindo comunicação entre aplicações ([12]). Esta comunicação é realizada através das trocas de mensagens, transmitidas em formato XML, incluindo parâmetros usados na chamada, bem como os dados de resultados. Isto significa que as mensagens podem ser entendidas por quase todas as plataformas de *hardware*, sistemas operacionais, linguagens de programação ou *hardwares* de rede ([61]). Também pode ser utilizado para invocar, publicar e localizar *Web Services* no registro UDDI.

O SOAP pode, potencialmente, ser utilizado em combinação com uma variedade de outros protocolos, como HTTP, SMTP, FTP, dentre outros, e suporta *remote procedure call* (RPC) ([37, 31, 32]).

O modelo de dados SOAP oferece definições para tipos de dados mais utilizados como *string*, *integer*, *float*, *double* e *date*. O processo de traduzir os dados (parâmetros e resultados) em XML é chamado codificação.

Um pacote SOAP consiste de quatro partes :

- Envelope SOAP: define um *framework* que indica o conteúdo da mensagem, quem poderá tratar esta mensagem e se o tratamento é opcional ou obrigatório. É uma estrutura de mensagem SOAP dentro do qual todos os outros elementos sintáticos da mensagem estão encapsulados;
- Codificação SOAP: define mecanismos de serialização que podem ser utilizados para trocar instâncias de tipos de dados definidos na aplicação;
- RPC SOAP: especifica como embutir *remote procedure call* e respostas dentro das mensagens com o propósito de invocar procedimentos em um sistema remoto;
- *Framework* de Ligação e Transporte SOAP: Define um *framework* abstrato para trocar Envelope SOAP entre pares usando um protocolo básico para transporte.

Na especificação do SOAP, outros conceitos também tornam-se importantes de serem mencionados ([33, 12]):

- Cliente SOAP*:* é um programa que cria um documento XML contendo a informação necessária para invocar remotamente um método em um sistema distribuído (também pode ser um servidor Web ou um servidor baseado em aplicações);
- Servidor SOAP: é responsável por executar uma mensagem SOAP e age como distribuidor e interpretador de documentos;
- Mensagem SOAP: é a forma básica de comunicação entre nodos SOAP;

```
<env:Envelope xmlns:env="">
  <env:Body>
    <m:getLastTradePrice
      env:encodingStyle="
http://www.w3.org/2001/06/soap- encoding"
      xmlns:m= "
http//www.w3.org/2001/06/quotes"
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </env:Body>
</env:Envelope>
```

FIGURA 2.5 – Exemplo de Código de Envelope SOAP ([12])

O formato de uma mensagem SOAP é um envelope contendo cabeçalhos opcionais e um corpo contendo uma mensagem atual com seus parâmetros ou resultados. As mensagens SOAP são escritas em XML. Na Figura 2.5 tem-se um exemplo de um código de uma mensagem SOAP.

O envelope SOAP apresenta a estrutura, conforme a Figura 2.6, onde pode-se vizualizar as partes que compõem o envelope. O bloco SOAP é uma construção sintática ou uma estrutura usada para delimitar dados que constitui, logicamente, uma única unidade computacional. O bloco é identificado por um elemento externo

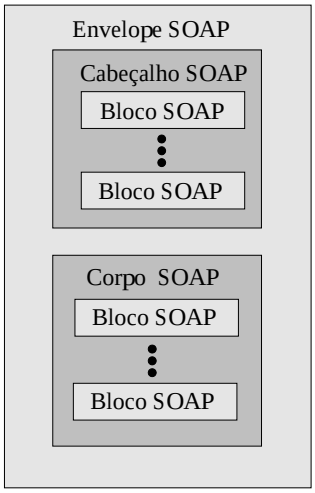

FIGURA 2.6 – Estrutura do Envelope SOAP ([12])

chamado *namespace* URI. O cabeçalho SOAP é uma coleção de zero ou mais blocos, os quais podem ser direcionados para um determinado receptor SOAP dentro do caminho da mensagem. O corpo SOAP é uma coleção de zero ou mais blocos direcionados para o último receptor SOAP.

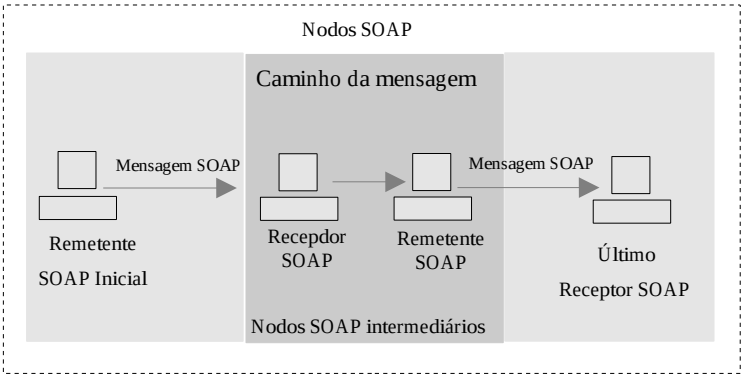

FIGURA 2.7 – Caminho das mensagens através dos nodos SOAP

A Figura 2.7 mostra como as mensagens trafegam através dos nodos SOAP. Como o SOAP não fornece roteamento, ele sabe qual remetente SOAP originou uma mensagem e quem será o último receptor a recebê-la, através de zero ou mais nodos intermediários.

Quando um nodo SOAP recebe uma mensagem, ele deve executar um processo, gerar uma mensagem de falha, gerar respostas e, se necessário, enviar mensagens adicionais.

A invocação do serviço utilizando SOAP ocorre conforme apresentado na Figura 2.8. A aplicação (1) requisita uma mensagem SOAP e invoca a operação do serviço através de um provedor de *Web Service*. O **Solicitante de Serviço** apresenta a mensagem junto com o endereço de rede do provedor de *Web Service*. A infra-estrutura de rede (2) entrega a mensagem para um servidor SOAP. O servidor

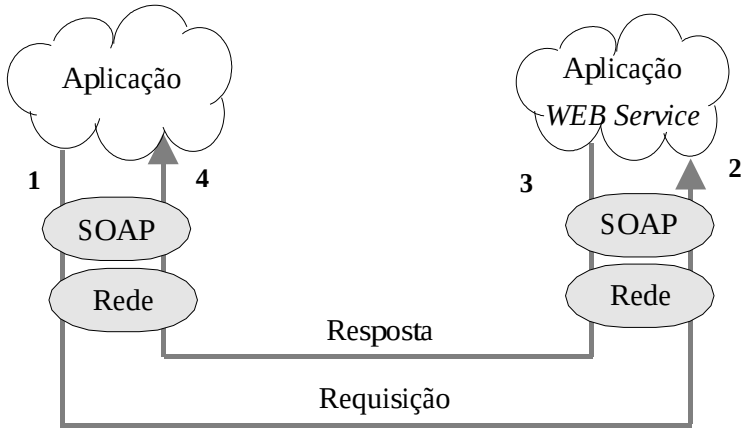

FIGURA 2.8 – Mensagens XML usando SOAP ([48])

SOAP redireciona a mensagem requisitada para o **Provedor de Serviço** *Web Service*. O servidor Web (3) é responsável por processar uma mensagem de requisição e formular uma resposta. A mensagem (4) é redirecionada através da infraestrutura SOAP. Quando a mensagem XML chega no nodo requisitante, é convertida para uma linguagem de programação, sendo então entregue para a aplicação.

Para que um serviço seja utilizado, faz-se necessário que o cliente saiba localizá-lo. Esta localização pode ser feita através da UDDI, que será apresentada na próxima seção.

2.3.3 *Universal Description, Discovery, and Integration* (UDDI)

A UDDI é um esforço para definir e criar um registro de serviços. Este registro pode ser acessado por clientes e estes poderão localizar todos os serviços que necessitem ([6, 24]).

O componente central chamado UDDI *Project* manipula um registro global público chamado UDDI *business registry*. Toda a informação mantida no registro público está disponível para consultas em geral. Um registro privado pode adicionar controle de segurança para proteger a integridade dos dados e previnir acessos não autorizados. Assim sendo, pode conter somente informação privada, conter um subconjunto de informação do registro público, ou ainda uma combinação destes. A informação oferecida pelo *business registry* consiste de três componentes:

- "páginas brancas" - endereço, contato e identificadores conhecidos;
- "páginas amarelas" - categorização industrial;
- "páginas verdes" - informações.

A implementação da UDDI é um registro *Web Service* que fornece um mecanismo para publicar e encontrar *Web Services*. Um registro UDDI contém informação categorizada sobre negócios, serviços que eles oferecem e associações destes serviços com especificações dos *Web Services*. Estas especificações normalmente são feitas em WSDL através de um registro UDDI ([58]).

O modelo de informação principal usado pelo registro UDDI é definido através de um esquema XML. O esquema XML define 4 tipos de informações:

- informações do negócio;
- informações do serviço;
- informações de ligações;
- informações específicas para serviços.

A informação referente ao registro de um serviço consiste de cinco tipos de estrutura de dados ([24]). Esta divisão por tipos de informação fornece partições simples para auxiliar na rápida localização e compreensão das diferentes informações que compõem o registro. As cinco estruturas são exibidas na Figura 2.9.

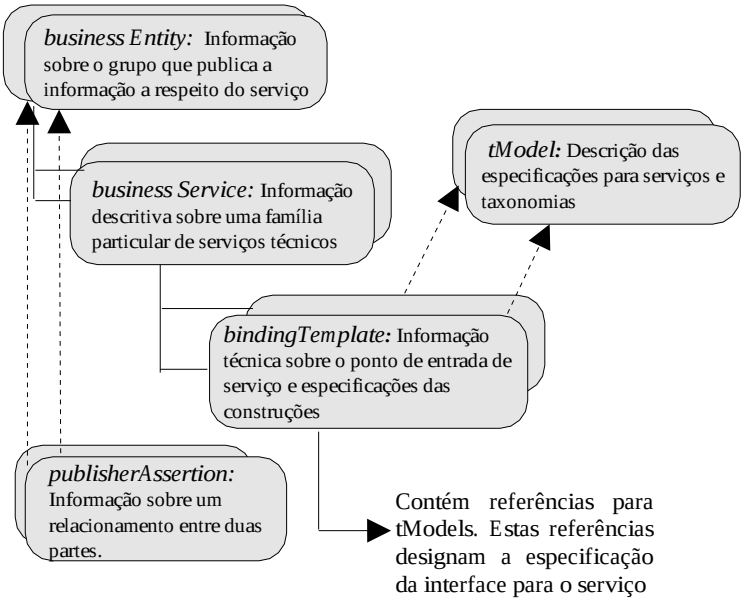

FIGURA 2.9 – Estrutura UDDI ([24])

O ***businessEntity*** fornece informação sobre um negócio, podendo conter um ou mais ***businessServices***. As descrições técnicas e de negócios para um *Web service* são definidas em um ***businessService*** e pelo seu ***bindingTemplate***. Cada ***bindingTemplate*** contém uma referência a um ou mais ***tModels***. Um ***tModel*** é usado para definir a especificação técnica de um serviço.

O *businessEntity*

Esta estrutura representa toda a informação conhecida sobre um negócio ou informações descritivas sobre uma entidade, bem como os serviços que ela fornece.

O *businessService*

Representa uma classificação lógica do serviço. O nome do elemento inclui o termo "*business*" numa tentativa de descrever o propósito deste nível na hierarquia de descrição de serviços. Assim, cada estrutura ***businessService*** é o filho lógico de uma única estrutura ***businessEntity***.

O *bindingTemplate*

As estruturas ***bindingTemplate*** são descrições técnicas de *Web Services* que estão acomodadas em instâncias individuais de ***bindingTemplate***. Essas estruturas fornecem suporte para acessar remotamente os serviços. O suporte a tecnologias, parâmetros específicos da aplicação e os arquivos de configuração também são encontrados.

O *tModel*

A estrutura **tModel** é representada por meio de metadados (dados sobre dados). O propósito de um ***tModel*** dentro de um registro UDDI é oferecer um sistema de referência para os documentos WSDL.

O *publisherAssertion*

Muitos negócios não são efetivamente representados por um único ***businessEntity***. Como conseqüência, diversas estruturas ***businessEntity*** podem ser publicadas. Mesmo assim, elas continuam representando um agrupamento e poderiam necessitar que alguns de seus relacionamentos fossem visíveis em seus registros UDDI. Quando dois negócios estão relacionados utilizam mensagens ***publisherAssertion***, como forma de publicar declarações a respeito das relações existentes entre eles. Ambos devem publicar exatamente a mesma informação, para que o relacionamento torne-se visível entre eles.

Um registro UDDI roda em um servidor UDDI. O registro é uma aplicação Web que pode se acessada pelo *browser* ou por uma API programável, através do protocolo SOAP.

2.3.4 Publicando e Encontrando Descrições de Serviço WSDL no UDDI *registry*

As entidades de dados UDDI fornecem suporte para determinar informações sobre negócios e serviços. A informação sobre a descrição de um serviço definida em WSDL é complementar à informação encontrada em um registro UDDI. A UDDI fornece suporte para diversos tipos de descrições de serviço. A UDDI não possui suporte direto a WSDL, ou a qualquer outro mecanismo de descrição de serviço ([14, 15]).

Uma descrição de serviço WSDL completa é a combinação de uma interface de serviço e um documento de implementação do mesmo. Esta implementação descreve as instâncias de um serviço. Cada instância é definida usando-se um elemento de serviço WSDL.

Uma implementação de serviço é publicada em um registro UDDI como um ***businessService*** com um ou mais ***bindingTemplates***. O ***businessService*** é

publicado pelo provedor de serviço.

Uma vez que a interface de um serviço representa uma definição reusável, esta interface pode ser publicada em um registro UDDI como um ***tModel***. Tal operação deve ser feita antes que uma implementação do serviço seja publicada como um ***businessService***.

As tecnologias padrões estão direcionando os esforços de algumas companhias em disponibilizar arquiteturas para desenvolvimento de *Web Services*.

2.4 Arquiteturas e Plataformas de Desenvolvimento de *Web Services*

A partir da arquitetura básica de *Web Services*, algumas companhias estão disponibilizando suas propostas bem como suas plataformas de desenvolvimento. As subseções seguintes apresentam as plataformas propostas pela IBM, HP SUN e Microsoft.

2.4.1 Modelo de *Web Services* proposto pela IBM

O modelo proposto pela IBM baseia-se na arquitetura geral de *Web Services* apresentada na Figura 2.1. A IBM desenvolveu o *WebSphere,* uma ferramenta destinada a aumentar a funcionalidade de um servidor Web padrão. A plataforma estende um ambiente de desenvolvimento adicionando suporte para *Java2 Enterprise Edition* (J2EE), CORBA, XML e serviços Web.

O *WebSphere Studio Application Developer* apresenta Java como a principal escolha para o desenvolvimento de *Web Services*. O *Application Developer* utiliza "*Web Services Wizards*" que facilitam o desenvolvimento de *Web Services* ([65]):

Web Service Wizard

O Web Service Wizard apresenta os passos conforme a Figura 2.10:

- Passo 1: a partir de um *JavaBean* existente é possível especificar quais métodos serão expostos como um *Web Service*. O assistente gera então os arquivos WSDL correspondentes. O assistente gera o arquivo ISD, que é o descritor de desenvolvimento SOAP para um *Web Service* em particular (dds.xml)

- Passo 1': é necessário também especificar a codificação SOAP, que diz ao ambiente de execução como traduzir estruturas de dados construídas em Java para SOAP XML e vice-versa.

- Passo 5: é possível gerar uma classe *proxy* do *Web Service*. Esta classe contém o código para configurar os estilos de codificação de parâmetros de entrada e saída. A classe *proxy* Java encapsula o código para fazer uma requisição SOAP e chamar o *Web Service*.

- Passo 6c: além disso, é esta classe que efetua a requisição SOAP.

- Passo 6b: o *wizard* permite gerar uma aplicação (JSPs) que invoca métodos do *Web Service* e exibe os resultados.

- Passos 6c e 6d: o assistente adiciona dois *servlets* à aplicação: o Apache RPC-SOAP e Apache Message SOAP. Estes *servlets* recebem as solicitações SOAP de um cliente SOAP (passo 6c), invocam um *Web Services* (passo 6d) e enviam uma resposta SOAP.

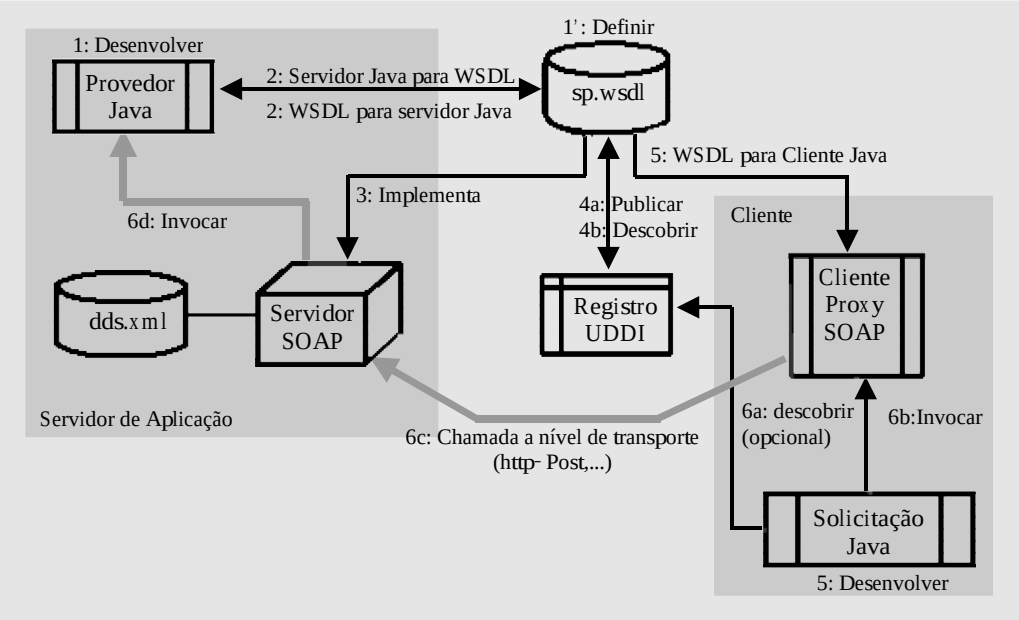

FIGURA 2.10 – Processo de desenvolvimento de um *Web Services* ([65])

Web Service Skeleton JavaBean Wizard

Este assistente gera um *JavaBean* a partir de um documento WSDL. O *JavaBean* contém definições de métodos conforme as operações descritas no documento WSDL. Com isso é possível implementar vários métodos a partir de uma estrutura básica, completando assim a implementação do *Web Service*.

Web Service DADX Group Configuration Wizard

O assistente permite criar um grupo *Document Access Definition Extension* (DADX). Um documento DADX especifica como criar um *Web Service* usando um conjunto de operações que são definidas por declarações SQL ou arquivos DAD. Os arquivos DAD descrevem o mapeamento entre os elementos de um documento XML e colunas de um banco de dados DB2. É utilizado pelo DB2 XML *extender* para armazenar e recuperar documentos XML de um banco de dados DB2 ([17]).

UDDI browser (import, export)

O *Application Developer* contém uma aplicação Web que forma o IBM UDDI *explorer*. Ele permite realizar as seguintes funções:

- Passo 4b: encontrar uma *businessEntity* e encontrar uma interface e a implementação do serviço;

- Passo 4a: publicar uma *businessEntity* e publicar uma interface e a implementação do serviço.

2.4.2 Estrutura da arquitetura do Modelo HP

A partir dos padrões já existentes para a arquitetura de *Web Services*, a Figura 2.11 apresenta elementos conhecidos como:

- Registro de Serviço que fornece meios para publicar e localizar *Web Services*;
- O Usuário do Serviço que utiliza os serviços disponíveis;
- O Provedor de Serviço disponibilizando e publicando uma aplicação como um serviço.

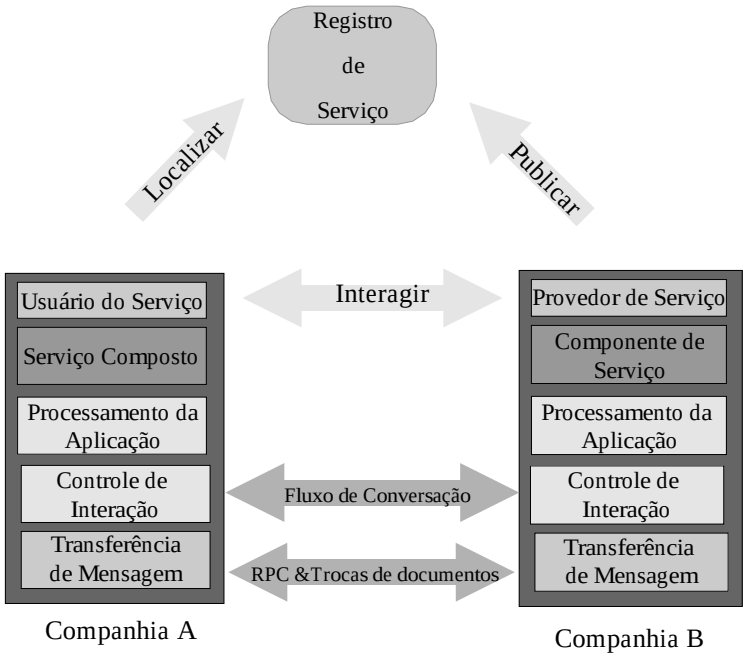

FIGURA 2.11 – Estrutura *Web Services* HP ([39])

Além destes elementos, a plataforma HP suporta interações de *Web Services* através de uma arquitetura que abrange três áreas chaves como: transferência de mensagem, controle de interação e processamento de aplicação. Com base nesta estrutura, o modelo HP fornece padrões para criar, disponibilizar e consumir serviços Web. Os núcleos principais do HP *Web Service* são XML e Java. Também utiliza padrões emergentes como UDDI, SOAP e WSDL, além de fornecer ferramentas de suporte e desenvolvimento ([39]).

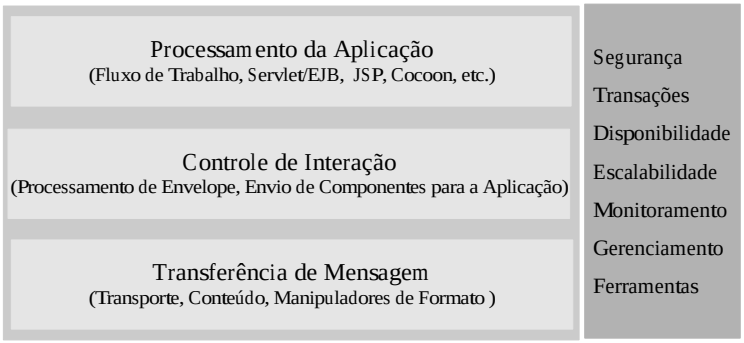

FIGURA 2.12 – HP *Web Services* Plataforma 1.0 ([39])

O modelo HP suporta interações de *Web Services* através de uma arquitetura modular que pode ser vista na Figura 2.12 e será descrita sussintamente:

- **Transferência de Mensagens:** suporta protocolos de comunicação de baixo nível entre os *Web Services* participantes. Fornece transformação e encaminhamento de mensagens para componentes de processamento e formatos alternativos (não-XML). Disponibiliza suporte para RPC-SOAP baseado em mensagens síncronas sobre HTTP/HTTPS, bem como processamento de mensagens em formato XML e transformação de conteúdo, incluindo suporte para SAX, DOM e XSTL.

- **Controle de Interação:** gerencia a troca de mensagem no contexto de negócio, realizando o processamento de alto nível no protocolo. Gerencia também a validação de mensagens e conversação. Disponibiliza processamento de envelope SOAP versão 1.1, além de requisitar suporte para envio, incluindo adaptadores à invocação de componentes lógicos J2EE e sistemas de comunicação para fluxo de trabalho.

- **Processamento da Aplicação:** habilita a lógica de negócio que é a essência do *Web Service*. Esta lógica pode estar contida em bancos de dados e sistemas corporativos. Amplia o motor básico J2EE. Faz integração com banco de dados e sistemas legados. Também habilita o uso de serviços baseados em Cocoon2[1] para invocar outros *Web Services*.

Além disso, são apresentadas nesta arquitetura, ferramentas de desenvolvimento, entrega e configuração relacionadas abaixo:

- Ferramentas para navegar em registros UDDI e publicar descrições de *Web Services* para UDDI.

- Fornece um registro UDDI privado para utilização durante o desenvolvimento.

- Disponibiliza suporte para mensagens SOAP e registros UDDI baseados em Cocoon2 que habilita o desenvolvimento de componentes que acessam outros *Web Services*.

[1] Cocoon2 é um *framework* para processar documentos XML. A plataforma HP *Web Services* usa este *framework* para entregar os resultados para o *browser* em HTML, WML ou outro formato específico.

- Provê um gerador *proxy* que oferece suporte à invocação de *Web Services* por clientes remotos.
- Suporta o uso de *Integrated Development Environment* (IDEs) populares para desenvolvimento de componentes para *Web Services*.
- Fornece ferramentas e *scripts* de configuração que habilitam a adaptação da plataforma às necessidades do cliente.

2.4.3 Estrutura do Modelo Sun

A arquitetura de *Web services* da Sun está dividida em três componentes lógicos, apresentados na Figura 2.13. O **Solicitante de Serviços** possui a capacidade de entregar serviços, permitindo servir pedidos de agentes ou clientes. É possível fazer uma verificação no **Serviço de Diretório** *(Service Broker)*, que contém metadados sobre os *Web Services* registrados, para fazer as ligações para a interface do serviço e invocar serviços dos **Provedores de Serviços**.

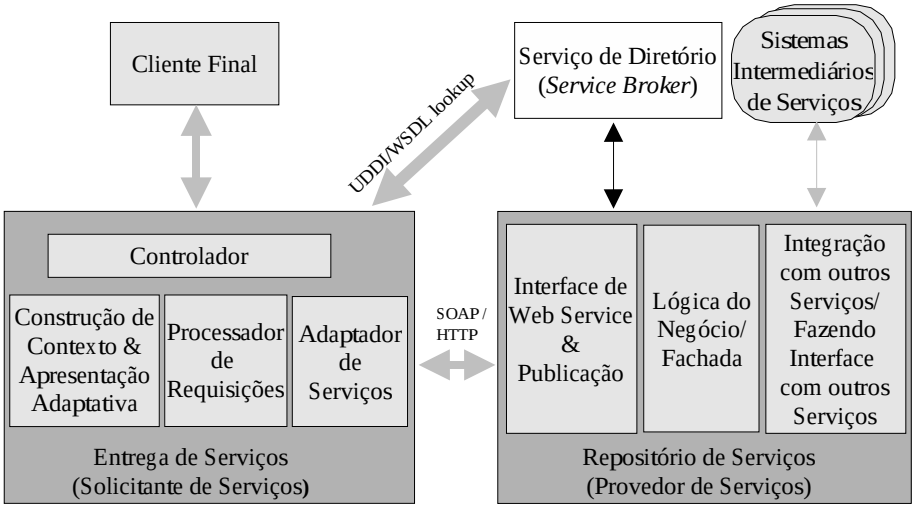

FIGURA 2.13 – Como os componentes interagem ([67])

Todas essas atividades podem ser modeladas em uma interface chamada **Processador de Serviços** ou em um módulo abstrato, com múltiplos processadores tais como: processador de e-mail, processador de localização e processador de impressão, para realizar serviços básicos de procura/ligação/invocação ([67]).

Em adição aos componentes acima, há também o componente **Controlador**. Ele é usado para a coordenação de interações e a invocação de serviços, bem como o gerenciamento geral dos processos.

A partir desta proposta de arquitetura, a Sun apresenta na plataforma *Java 2 Plataform, Enterprise Edition* (J2EE), que fornece componentes para a construção de *Web Services*. A Figura 2.14 mostra o desenvolvimento de *Web Services* utilizando J2EE, onde observam-se algumas formas utilizadas por clientes para acessar serviços. Estes acessos podem ser feitos através de *Web Browser* (HTTP), de alguma aplicação ou *applets* (IIOP). Clientes podem por exemplo, conectar-se a bases de dados utilizando *Java Database Connectivity* (JDBC) ou SQL. Os serviços também

podem ser acessados por clientes que utilizem tecnologias *Web Services* baseadas em SOAP, UDDI, WSDL e ebXML ([2]).

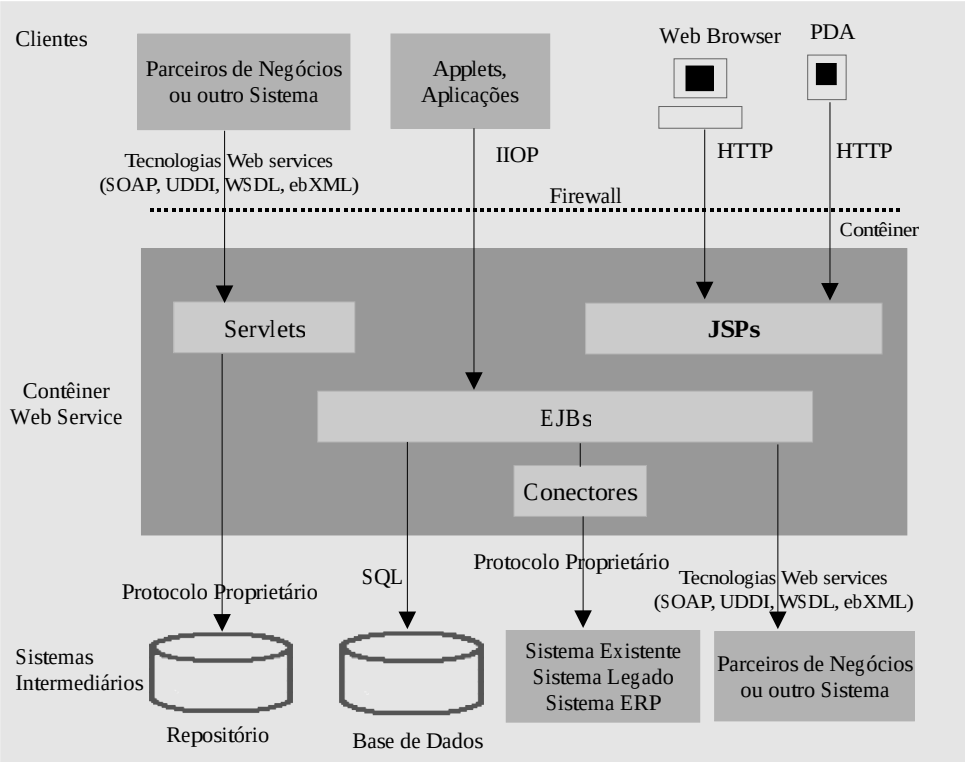

FIGURA 2.14 – Desenvolvimento de *Web Services* com J2EE ([46])

Para que estes relacionamentos se tornem possíveis, o modelo de desenvolvimento J2EE baseia-se nas tecnologias XML e Java. Desta forma, a Sun disponibiliza uma plataforma de desenvolvimento Java *Web Services Developer Pack* (Java WSDP) a qual apresenta um conjunto de APIs Java para XML, *servlet* Tomcat e *container* JSP, bem como um servidor de registros WSDL. Através das APIs torna-se possível acessar documentos XML e realizar operações. Estas APIs são descritas brevemente a seguir ([2, 46]):

- Java API para processamento XML (JAXP) - é uma API com o objetivo de acessar, modificar e criar documentos XML em Java. Processa documentos XML utilizando vários *parsers* como *Simple API for XML parsing* (SAX) e *Document Object Model* (DOM). Também suporta XML *Stylesheet Language Transformations* (XSLT), dando controle para habilitar e converter dados a documentos XML ou outros formatos.

- Java API para mensagens XML (JAXM) - envia mensagens SOAP sobre a internet em um modo padrão. Pode ser estendida para trabalhar com protocolos de mensagem como ebXML.

- Java API para registros XML (JAXR) - apresenta um caminho padrão para acessar registros de negócios e informações relacionadas. Permite escrever aplicações em linguagem de programação Java fornecendo um caminho uniforme de acesso a registros baseados em padrões abertos como ebXML ou UDDI.

- Java API para XML baseada em RPC (JAX-RPC) - torna possível escrever uma aplicação em linguagem de programação Java que utiliza SOAP para fazer uma *Remote Procedure Call* (RPC). Também pode ser utilizada no envio de mensagens de requisição e resposta. Torna possível uma aplicação definir seu próprio esquema XML e utilizá-lo para enviar documentos e fragmentos XML.

O uso destas APIs permite que seja possível acessar serviços baseados em XML utilizando linguagem de programação Java conforme, a Figura 2.15.

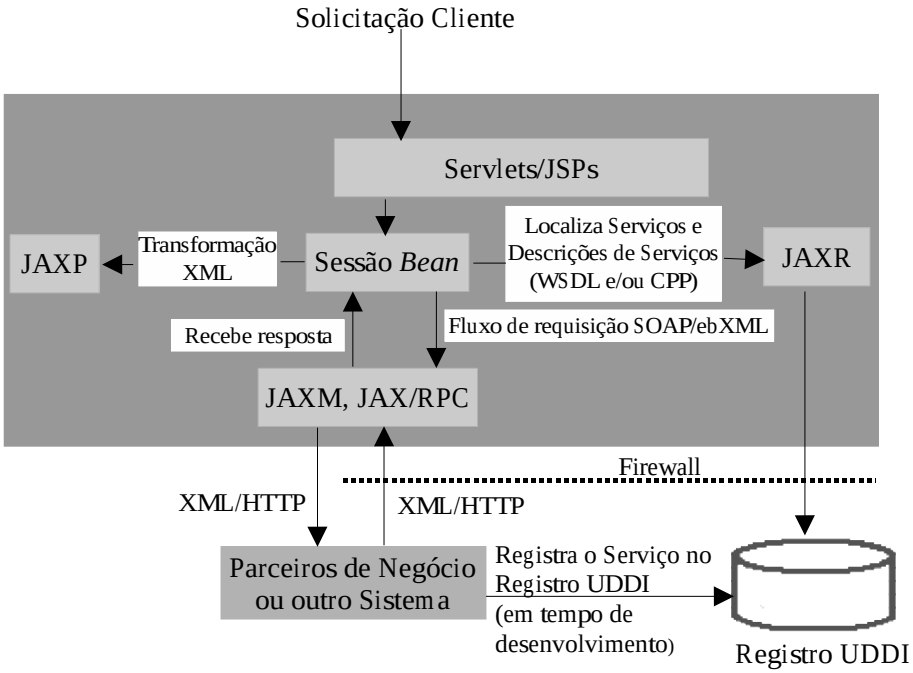

FIGURA 2.15 – Usando as APIs JAX para invocar *Web Services* *([46])*

2.4.4 O modelo proposto pela Microsoft

A plataforma Microsoft .NET suporta a arquitetura XML *Web Services* padrão. A plataforma e tecnologias .NET apresentam um conjunto de novos produtos que baseiam-se no uso de XML, dando uma grande ênfase ao protocolo SOAP ([35]).

Desta forma, os dados podem ser facilmente organizados, programados, editados e trocados entre *Web Sites*, aplicações e dispositivos. Na Figura 2.16 observamos que a comunicação entre diferentes *Web Services* é feita utilizando este padrão de comunicação ([66, 53]).

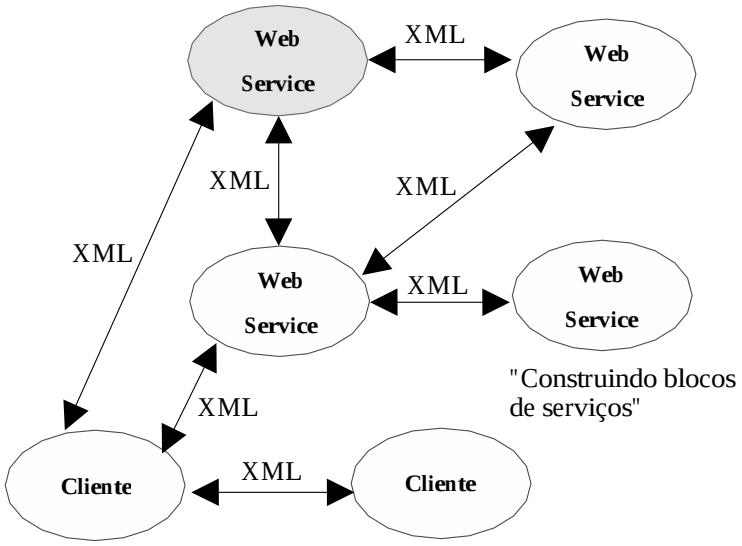

FIGURA 2.16 – Modelo *Web Services* ([53])

O XML *Web Services* proposto pela Microsoft, consiste em blocos de construção para facilitar a computação distribuída sobre a Internet. O uso de padrões abertos focados na comunicação e colaboração entre pessoas e aplicações têm criado um ambiente onde os XML *Web Services* estão se tornando uma plataforma para integração de aplicações. Aplicações são construídas usando múltiplos XML *Web Service*s de várias fontes que trabalham juntas, independente de onde estejam ou de como foram implementadas.

A Figura 2.17 apresenta a forma utilizada por uma aplicação para fornecer dados e serviços a outras aplicações e usuários, através da Internet, uzando-se uma via comum de entrega. As aplicações acessam *Web Services* via protocolos Web (presentes em todos os lugares) e formatos de dados como XML, UDDI e SOAP, sem a necessidade de preocupar-se como cada *Web Service* é implementado.

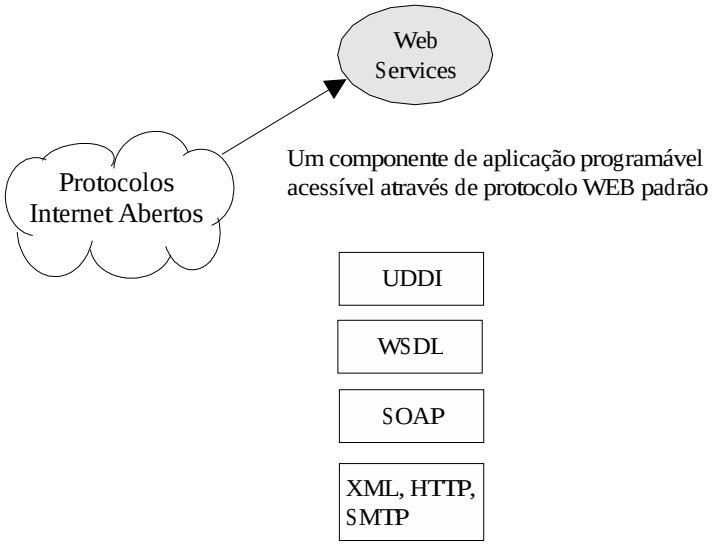

FIGURA 2.17 – *Web Services* ([53])

A plataforma .NET habilita a criação e uso de aplicações, processos e sites Web baseados em XML, como serviços que compartilham e combinam informações e funcionalidades com outras plataformas ou dispositivos ([53]).

Os componentes para desenvolvimento de *Web Services* da plataforma .NET são os seguintes ([47]):

- ASP+ *runtime*: representa uma abstração na programação que permite aos desenvolvedores construir *Web Services* a partir de uma estrutura pré-pronta.
- Biblioteca de classes .NET: Oferece um conjunto de classes para construir e manipular *Web Services*.
- *Web Services Description Language Tool*: gera um código de serviços para XML *Web Services* e serviços Web clientes a partir de arquivos WSDL e arquivos *XML Schema Definition* (XSD).
- *Web Services Discovery Tool*: Descobre as URLs de XML *Web Services* localizados em um servidor Web e grava documentos relacionados a cada XML *Web Services* localmente.
- Web *References*: É uma interface gráfica que incorpora a funcionalidade do *Web Service Description Language Tool*.
- *Server and Solution Explorer Tool*: É uma interface gráfica para navegar em um *Web Services* e gerenciar projetos de *Web Services*.

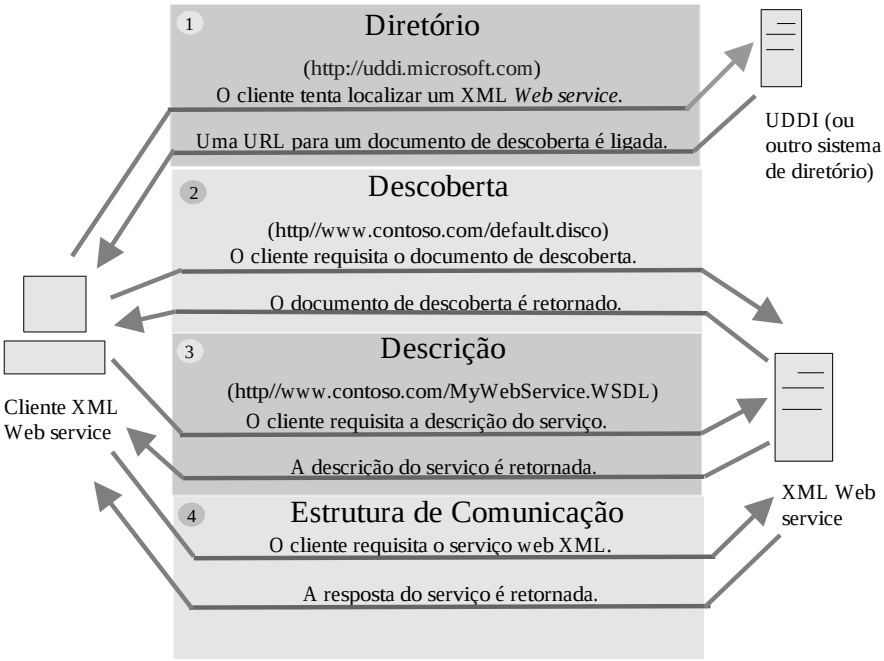

FIGURA 2.18 – Infraestrutura XML *Web Services* no .NET ([47])

A infraestrutura da Microsoft para XML *Web Services* ilustrada na Figura 2.18 apresenta três entidades ou papéis: O XML *Web service client*, o XML *Web*

service e o *Directory Service*. A seguir, as operações serão descritas através de um exemplo:

1. O cliente está interessado em um serviço de verificação de cartão de crédito. A UDDI provê um diretório de serviços e funcionalidades de busca através de uma API padronizada.

2. Um cliente requisita um documento de descoberta do serviço, o qual reside, por convenção, no diretório raiz do servidor Web. Um documento de descoberta contém ponteiros para descrições de serviços e localizações. Caso o cliente já saiba a URI de um serviço Web, este passo pode ser omitido.

3. O *Web Service* cliente requisita então uma descrição do serviço, na forma de um documento WSDL através de uma URL conhecida, usando HTTP. O documento WSDL retornado contém todas as informações técnicas necessárias para invocar o serviço Web (lista de parâmetros e tipos de dados).

4. O cliente faz uma requisição para um XML *Web Service* usando SOAP para transmitir uma mensagem que contém os nomes dos métodos e os seus parâmetros. O XML *Web Service* processa a requisição e retorna um envelope SOAP que contém os resultados computados.

Existem diversas definições de XML *Web Services,* porém todas definições têm alguns itens em comum:

- XML *Web Services* mostram funcionalidades úteis para usuários através de um protocolo Web padrão. Na maioria dos casos o protocolo usado é SOAP.

- XML *Web Services* fornecem uma maneira de descrever suas interfaces com detalhes suficientes, a ponto de permitir que um usuário construa uma aplicação cliente para falar com eles. Esta descrição geralmente é fornecida em um documento XML chamado WSDL.

- XML *Web Services* são registrados de forma que os usuários possam encontrá-los facilmente. Isto é feito por meio de UDDI.

A seguir, uma comparação das principais abordagens já mencionadas.

2.4.5 Um Comparativo entre as Arquiteturas

Considerando-se as informações dispostas a respeito das arquiteturas propostas, nesta seção faz-se uma breve análise visando uma melhor compreensão das tecnologias e padrões utilizados.

	.NET (Microsoft)	J2EE (Sun)	IBM *Web Services*	HP *Web Services*
Tecnologia	.NET	J2EE	J2EE	J2EE
Ambiente de Desenvolvimento	Visual Studio.NET	J2EE+JWSDP	J2EE+WebSphere	J2EE+Cocoon
Plataforma de Servidor	Windows NT/.NET	JVM *Complient*	JVM *Complient*	JVM *Complient*
Registro de Serviços	UDDI ou outro Sistema de Diretório	UDDI	UDDI	UDDI
Descoberta de Serviços	WSDL (arquivo DISCO)	WSDL	WSDL	WSDL
Formato de Comunicação	SOAP	SOAP	SOAP	SOAP

TABELA 2.1 – Comparativo entre arquiteturas de *Web Services* ([34])

A Tabela 2.1 apresenta um comparativo entre as tecnologias empregadas nas arquiteturas propostas pela Sun, Microsoft, IBM e HP. O uso de tecnologias padrões como SOAP, WSDL e UDDI pelas diferentes companhias tem como principal objetivo a interoperabilidade entre diferentes plataformas. Pode-se observar, no entanto o uso de duas tecnologias concorrentes tais como J2EE e .NET.

A tecnologia J2EE é adotada pela maioria das corporações que estão direcionando seus esforços para o desenvolvimento de arquiteturas de *Web Services*.

Características	.NET	J2EE
Tipo de Tecnologia	Produto	Padrão
Soluções Intermediárias	Microsoft	30+
Interpretador	CLR	JRE
Páginas Web Dinâmicas	ASP.NET	JSP
Componentes de camada Intermediária	.NET Componentes de Gerenciamento	EJB
Acesso a Base de Dados	ADO.NET	JDBC,SQL/J
SOAP, WSDL, UDDI	Sim	Sim
Suporte Implícito de Arquiteturas (balanceamento de carga, etc)	Sim	Sim

TABELA 2.2 – Comparativo entre as Tecnologias J2EE e .NET ([64])

Por outro lado, a tecnologia proposta pela Microsoft tende a integrar a arquitetura de *Web Services* a uma arquitetura proprietária. A partir das duas tecnologias destacadas, há na Tabela 2.2 um comparativo entre elas. É possível observar as

características destas duas tecnologias em relação a vários pontos como interpretadores utilizados, servidores de páginas Web dinâmicas, forma de acesso a bases de dados dentre outros.

2.5 Conclusão

Este capítulo apresentou as principais abordagens de arquiteturas de *Web Services,* bem como as tecnologias padrões que estão sendo utilizadas. As diferentes abordagens de arquiteturas demonstram que padrões como SOAP, UDDI e WSDL representam uma forma de permitir a comunicação de *Web Services* construídos em diferentes plataformas.

Observa-se, também, que companhias como Sun, IBM e HP estão direcionando seus esforços de forma a não deixar a plataforma de desenvolvimento comprometida a uma arquitetura proprietária, ao contrário da plataforma Microsoft .NET.

O esforço comum direcionado para o uso de padrões baseados em XML, permite que aplicações possam se comunicar utilizando tecnologia de mensagem padrão. Desta forma, a linguagem de programação na qual a aplicação foi construída torna-se transparente.

Outra característica das arquiteturas de *Web Services*, é a possibilidade de construir novas aplicações utilizando-se a composição de serviços que estejam publicados em *Web Services*.

No próximo capítulo far-se-á um estudo sobre metadados educacionais. Os metadados educacionais possibilitam que se possa fazer descrição de recursos educacionais, de forma a permitir o compartilhamento dos mesmos. Também, será feito um breve estudo de algumas arquiteturas propostas para utilização de metadados educacionais.

Capítulo 3

Metadados Educacionais

Resumo

Este capítulo mostra como os objetos educacionais são modelados e quais são as principais arquiteturas existentes. Para isso, serão apresentadas algumas iniciativas de padronização de metadados, bem como as contribuições de alguns projetos para que esta padronização se torne possível.

3.1 Introdução

O uso de metadados permite a especificação de recursos computacionais, descrevendo suas características e estruturas, e também informações sobre sua acessibilidade.

As iniciativas de padronização de metadados têm como objetivo tornar os recursos interoperáveis. Além disso, algumas extensões de metadados estão sendo utilizadas em sistemas educacionais. Desta forma, os metadados educacionais permitem a interoperabilidade de conteúdos entre ambientes de aprendizagem.

3.2 O que são metadados?

Os metadados são "dados sobre dados" ou "informações sobre informações". São um conjunto de palavras, frases ou sentenças que resumem ou descrevem o conteúdo de um *site* Web ou uma página Web individual, um recurso computacional com o objetivo de beneficiar o trabalho de agentes de busca ([11, 21]).

Os metadados são descritos em formato XML, formando uma estrutura de dados que descreve as características de um recurso. Desta maneira, podem ser utilizados para descrever conteúdos e estruturas, além de fornecer informação sobre acessibilidade, organização e relacionamento entre os dados ([51, 59]).

Os sistemas educacionais podem obter benefícios com uso de metadados, porque estes propiciam o desenvolvimento de aplicações interoperáveis. A troca de recursos de aprendizagem entre os sistemas pode ser feita, mediante uma descrição formal das suas interfaces.

A padronização de metadados educacionais torna-se necessária por duas razões principais:

- Recursos educacionais são definidos, estruturados e apresentados em vários formatos.
- Módulos funcionais embutidos em um sistema de aprendizagem particular não podem ser facilmente reutilizados por outro sistema.

Alguns padrões de metadados para *e-learning* estão emergindo. Dentre eles: *Dublin Core* e IEEE-LOM. As principais iniciativas de padronização educacional são apresentadas na Tabela 3.1. Pode-se destacar iniciativas como: IEEE-*Learning Technology Standards Committee* (LSTC), ARIADNE, IMS, GESTALT dentre outras. Nas seções seguintes serão apresentados alguns metadados, bem como algumas das iniciativas de padronização permitindo uma visão geral de suas arquiteturas.

Sigla	Organização	Iniciativa
IEEE- LTSC	IEEE	Learning Technologies Standardization Committee
JTC12SC36	ISO and IEC	Joint Committee for the Standardization of Learning Technologies
IMS	EDUCASE	IMS Project &Consortion
AICC	US AI	Aviation Industry CBT Committee
ADL	US DoD	Advanced Distributed Learning
DC- ED	DCMI	Dublin Core Educational Metadata
GEM	US DoE	Gateway to Educational Materials
NSDL Metadata	NSDL SWG	National Science. Mathematics. Engineering and Technology Education Digital Library
ARIADNE	EC	Alliance of Remote Instructional Authoring Across Leading in European Society
GESTALT	EC	Getting Educational Systems Talking Across Leading and Training in European Society
PROMETEUS	EC	PROmoting Multimedia access to Education and Training in European Society
CEN/ISSS/LT	CEN	Learning Technologies Workshop
EdNA		Education Network Australia

JTC: Joint Technical Committee; SC: Subcommittee; DoD: Department of Defense; DoE: Department of Education; AI: Aviation Industry: ASDL SWG: NDSL Standards Working Group; DCMI: Dublin Core Metadata Iniciative; EC: European Community; European Standardization Committee.

TABELA 3.1 – Principais Iniciativas de Padronização Educacional ([3])

3.3 *Dublin Core*

O padrão *Dublin Core* é um formato que descreve recursos computacionais. A iniciativa *Dublin Core Metadata* (DCMI) define um caminho padrão para qualificar elementos. Apresenta duas classes de qualificadores ([38]):

- Refinamento de elemento: torna o significado de um elemento mais específico.
- Esquema de codificação: inclui um vocabulário controlado, notação formal ou regras de análise gramatical.

O padrão de metadados *Dublin Core* é constituído por quinze elementos divididos em três categorias que são ([21]):

- conteúdo;

- propriedade intelectual;
- instanciação.

Existem algumas iniciativas de extensão do *Dublin Core* para descrever recursos educacionais, adicionando elementos como: audiência, padrões, qualidade, tipo de interatividade, nível de interatividade e tempo de aprendizado. Na próxima seção algumas destas iniciativas.

3.3.1 Iniciativas GEM, NSDL e EdNA

O Projeto *Gateway to Educational Materials* (GEM) [29], provê uma estrutura para publicação e localização de recursos educacionais, disponíveis através da Internet. Esta ferramenta ajuda a organizar o conteúdo em um catálogo de recursos tornando-os disponíveis ao público. Estes recursos podem ser acessados através do *gateway* GEM. Este catálogo utiliza um metadado baseado em *Dublin Core*. O metadado GEM é composto por vinte e cinco elementos dos quais quinze provêm do *Dublin Core*, além de um vetor de controle de vocabulário. Os oito elementos adicionados para o *Dublin Core* representam informações importantes para que professores e educadores realizem busca por currículos, planos de lições e outros materiais educacionais.

O programa *National SMETE Digital Library* (NDSL) [55], tem como objetivo definir uma biblioteca digital que irá fazer parte de um ambiente de aprendizado on-line. O conjunto de metadados padrão proposto pelo NDSL tem vinte elementos e também está baseado no *Dublin Core*, o qual contribuiu com quinze elementos.

A iniciativa da *Education Network Australia* (EdNA) [23], tem como objetivo promover a Internet como uma ferramenta de suporte para aprendizado, através de recursos computacionais entre a comunidade européia. O principal objetivo é oferecer acesso a serviços e recursos educacionais. O EdNA define um esquema de metadados baseado no *Dublin Core*, estendendo-o em oito novos elementos que descrevem informações sobre classificação de recursos educacionais disponíveis.

A próxima seção apresenta outro padrão de metadados: o *Learning Object Metadata* (LOM).

3.4 Learning Object Metadata (LOM)

LOM especifica um esquema conceitual de dados o qual define a estrutura de uma instância de um metadado para objetos de aprendizagem. Neste padrão um objeto de aprendizagem é definido como uma entidade que pode ter um formato digital ou não. Este padrão pode ser utilizado para sistemas de aprendizado, educação e treinamento. Uma instância de um metadado descreve características relevantes do objeto de aprendizagem para os quais este possa ser aplicado ([59, 11, 40]).

Este padrão facilita a busca, a avaliação e o uso de objetos de aprendizagem por aprendizes, instrutores ou por ferramentas de software automatizadas. Também pode auxiliar no compartilhamento e a troca de conteúdos educacionais.

O LOM é caracterizado por permitir a definição de blocos independentes de conteúdo educacional. Estes blocos podem conter referências para outros objetos e podem ser combinados ou seqüenciados para formar grandes unidades educacionais ([40]).

A estrutura padrão do LOM é apresentada por um esquema básico formado por nove categorias ([40]):

- Geral: são as características que descrevem o objeto de aprendizagem como um todo.
- Ciclo de vida: define o estado corrente e quem usou o objeto de aprendizagem durante sua evolução.
- Meta-metadado: são as informações sobre a própria instância do metadado.
- Técnico: define os requisitos técnicos e características do objeto de aprendizagem.
- Educacional: apresenta as características pedagógicas.
- Direitos: define os direitos de propriedade intelectual e condições de uso do objeto de aprendizagem.
- Relação: define o relacionamento entre objetos de aprendizagem.
- Anotação: são as informações de quando e por quem o comentário foi criado.
- Classificação: descreve o objeto de aprendizagem, conforme um sistema de classificação particular.

Um dos principais contribuidores para a padronização dos metadados educacionais é o IEEE-LTSC. Em 1998 os projetos IMS e ARIADNE submeteram uma proposta de padronização que resultou no documento base IEEE-LOM. A seguir, apresentam-se iniciativas para padronização e extensão do metadado LOM.

3.5 O Projeto ARIADNE

O principal objetivo do projeto ARIADNE é promover busca e reuso de material pedagógico em formato eletrônico. Para alcançar este objetivo está sendo construído um grande repositório de documentos pedagógicos chamado *Knowledge Pool System* (KPS). Este sistema de repositório, apresentado na Figura 3.1, é composto por ferramentas de indexação e algumas ferramentas de consulta ([26]).

3.5.1 Sistema de Repositório ARIADNE

No sistema de repositório de documentos pedagógicos KPS, o componente **Ferramenta de Indexação**, é um gerador de cabeçalho pedagógico o qual é utilizado para criar e validar as descrições de documentos pedagógicos. O documento original, juntamente com a sua descrição, compõem o que é chamado pelo sistema de elemento pedagógico.

O componente **Ferramentas de Consulta** é composto por uma ARIADNE *Management Interface* (AMI), uma ARIADNE *Learner Interface* (ALI) e um editor de currículo. O editor de currículo é uma ferramenta para criação de cursos direcionados, ou seja, um currículo define a estrutura do curso através do número de sessões que são executadas em um determinado momento. O currículo também

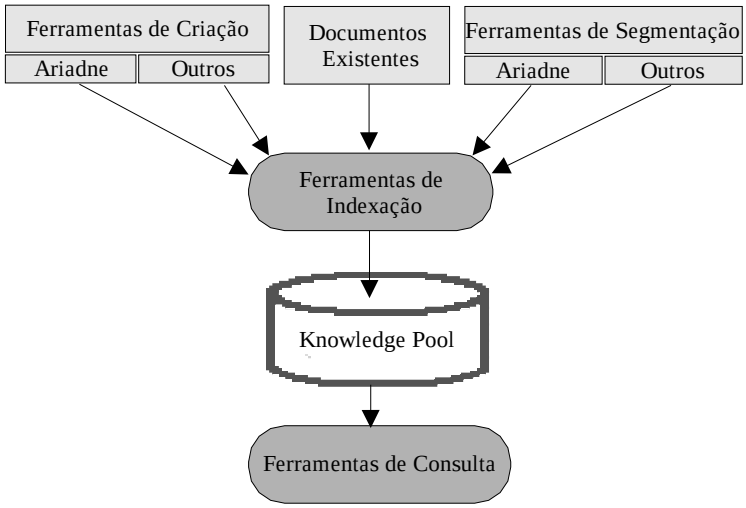

FIGURA 3.1 – Arquitetura do Sistema de Repositório ARIADNE ([22])

define que documentos do KPS estão sendo disponibilizados para os estudantes e em que sessões.

Com base no KPS, o projeto ARIADNE está desenvolvendo um ambiente de aprendizagem baseado na Web, conforme a Figura 3.2. Neste ambiente um educador pode criar e manter o currículo pedagógico, incorporando os objetos de aprendizagem do KPS e distribuir este currículo para estudantes com um mínimo de esforço ([22]).

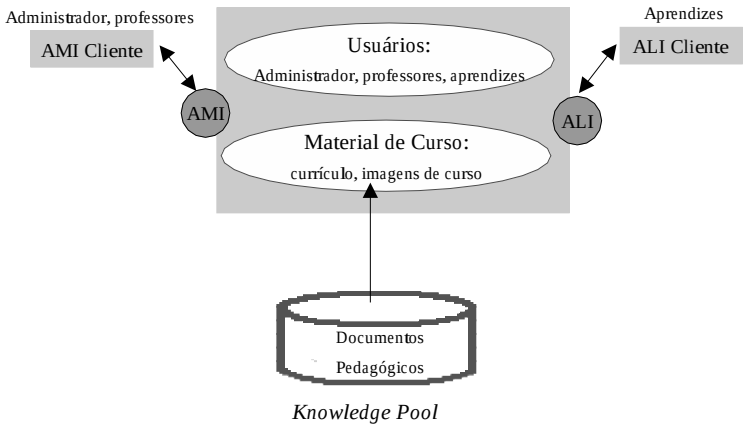

FIGURA 3.2 – O Ambiente de Aprendizagem Web ARIADNE ([22])

O AMI, que gerencia a interface do ARIADNE, permite que os educadores editem os cursos, transformando estes em cursos para Web, além de fazer manutenção e disponibilizá-los para estudantes. A disponibilização ocorre através do ALI. Desta forma, utilizando um *browser*, os estudantes são guiados através de materiais e informações relevantes.

Nesta arquitetura uma descrição de curso no formato ARIADNE inclui informações gerais do curso tais como: título, sumário e uma visão geral de todas as atividades ali presentes. A tradução de um currículo para um *site* Web é realizada automaticamente por meio de um mecanismo de modelos.

O uso de AMI e ALI, juntamente com KPS resulta em um mecanismo de comunicação de via única (educadores para estudantes). De maneira a acrescentar funcionalidades de interatividade, a plataforma de ensino ARIADNE pode ser estendida através de componentes externos ou módulos.

Uma das características deste repositório é a especificação básica dos metadados utilizados. O interesse desta iniciativa está centrado no desenvolvimento de um sistema de metadados que trabalhe com ambientes multilingüistas e multiculturais. O esquema de metadados utilizado está baseado em LOM ([26, 51]). Este esquema é utilizado pelas ferramentas de indexação para gerar o cabeçalho pedagógico do sistema. Este cabeçalho pedagógico como dito anteriormente é a descrição dos recursos pedagógicos armazenados no sistema KPS.

3.5.2 Esquema de Metadados ARIADNE

O conjunto de descritores no metadado educacional do ARIADNE [4, 5], está organizado em categorias, classificados como obrigatórios ou opcionais. Dentro da categoria dos elementos obrigatórios temos as seguintes classificações:

- informação geral do próprio recurso;
- semânticas do recurso;
- atributos pedagógicos;
- características técnicas;
- condições de uso;
- meta-metadado.

Dentro dos elementos opcionais temos a categoria:

- observações.

Outro projeto envolvido em padronizar os metadados educacionais é o projeto IMS. Além disso, este projeto tem sua atenção voltada à padronização de servidores de aprendizagem. Algumas das iniciativas deste projeto na seção seguinte.

3.6 O Projeto IMS

Fundado em 1997 e originalmente focado em educação superior, o projeto IMS teve seu foco direcionado para uma faixa de iniciativas relacionadas a padrões de servidores de aprendizado, conteúdo de aprendizado e integração dessas capacidades. O IMS *Global Learning Consortium* conta com a participação de diversas organizações tais como *Advanced Distributed Learning* (ADL), *Apple Computers*, *Artesia Technologies*, *California State University*, *Oracle*, *University of Cambridge*, *University of Michigan*, PUC-Rio dentre outras.

Algumas das atividades desenvolvidas e promovidas pelo projeto IMS para facilitar atividades de ensino on-line são ([51]):

- a localização e o uso de conteúdo educacional;

- o controle do progresso do aprendizado;
- o relatório da performance do aprendizado;
- a troca de registros de estudantes entre sistemas administrativos.

Já os principais objetivos do IMS são:

- a definição de técnicas padrões para a interoperabilidade de aplicações e serviços em aprendizado distribuído;
- o suporte à incorporação de especificações IMS em produtos e serviços;
- a adoção difundida das especificações que irão permitir ambientes de aprendizado distribuído e conteúdos de múltiplos autores a trabalharem em conjunto.

O projeto IMS apresenta algumas especificações importantes tais como ([45]):

- **IMS *Content Packaging Specification:*** descreve uma estrutura que torna possível a criação de objetos de conteúdo que são reutilizáveis em uma variedade de sistemas de aprendizagem.
- **IMS *Question & Test Interoperability* (QTI):** descreve uma estrutura básica para a representação de dados de questões e testes (avaliação), utilizando a adaptação da QTI para habilitar a troca de dados para teste e avaliação entre *Learning Management Systems* (LMS), assim como conteúdo de autores, bibliotecas e coleções.
- **IMS *Learning Resources Meta-Data Specification*:** cria uma maneira uniforme de descrever recursos educacionais, tal que estes possam ser encontrados mais facilmente usando uma ferramenta de busca baseada em meta-informação.
- **IMS *Enterprise Specification*:** descreve como sistemas de gerenciamento instrucional são interoperáveis com outros sistemas usados para suportar operações de uma organização assim como administração de treinamento, de estudantes e sistemas de recursos humanos.
- **IMS *Metadata Specification*:** torna o processo de encontrar e utilizar recurso de aprendizagem mais eficientes por fornecer uma estrutura de elementos definidos que descrevem ou catalogam um recurso de aprendizagem, juntamente com os requisitos sobre como os elementos podem ser trabalhados.
- **IMS *Acessibility Working Group*:** fornece referências e informa sobre acessibilidade à comunidade de aprendizagem distribuída para assegurar quais recursos podem ser acessados por qualquer entidade, a qualquer hora e em qualquer lugar.

Outras especificações estão em fase de desenvolvimento como a IMS *Learning Design Working Group* e IMS *Digital Repositories Working Group*.

A partir do padrão IEEE-LOM, o projeto IMS desenvolveu uma representação de metadado em XML ([43]). A Figura 3.3 apresenta um exemplo de recurso educacional descrito em LOM-XML.

```
<?xml version="1.0" encoding="ISO- 8859- 1" standalone="yes"?>
<lom>
 <metametadata>
  <language>pt- BR</language>
 </metametadata>
 <general>
  <title>
   <langstring lang="pt- BR">Tutorial Java</langstring>
  </title>
  <catalogentry>
   <entry>
    <langstring lang="pt- BR">Tutorais</langstring>
   </entry>
  </catalogentry>
  <language>pt- BR</language>
  <description>
   <langstring lang="pt- BR">Tutorial sobre a linguagem Java</langstring>
  </description>
  <keywords>
   <langstring lang="pt- BR">Java</langstring>
   <langstring lang="pt- BR">Tutorial</langstring>
   <langstring lang="pt- BR">JDK</langstring>
  </keywords>
  <coverage>
   <langstring lang="pt- BR">J2SE 1.4.1</langstring>
  </coverage>
  <structure>
   <langstring>Atomic</langstring>
  </structure>
  <aggregationlevel>0</aggregationlevel>
 </general>
 <lifecycle>
  <version>
   <langstring lang="pt- BR">1.0</langstring>
  </version>
  <status>
   <langstring>Final</langstring>
  </status>
  <contribute>
   <role>
    <langstring>Author</langstring>
   </role>
   <centity>
    <vcard>Roseli Persson Hansen</vcard>
   </centity>
```

FIGURA 3.3 – Exemplo de LOM-XML

Como o LOM não captura adequadamente todos os metadados necessários para descrever objetos de aprendizagem e respectivos usos, ele permite extensões para elementos e estruturas de metadados proprietárias. Entretanto o abuso de elementos de dados pode comprometer a interoperabilidade semântica. Para evitar esta quebra de interoperabilidade, o IMS definiu duas maneiras de tratamento para todas as extensões proprietárias ([42, 41]):

- **Extensões usando *Document Type Definition* (DTD):** o IMS definiu um elemento que deve ser usado quando deseja-se construir extensões usando um arquivo de controle DTD. Este elemento é opcional para cada parte da estrutura de metadados. Utilizar extensões com mais de um arquivo de controle DTD é problemático. O IMS recomenda utilizar a linguagem XML *Schema*

Definition quando usar extensões.

- **Extensões usando XML *Schema Definition* (XSD):** o suporte à extensão de metadados usando a linguagem XML *Schema Definition* é possível através do uso de outros elementos LOM como novos elementos definidos em um *namespace* específico.

O projeto IMS apresenta uma tecnologia de repositório para suportar a configuração, apresentação e entrega de objetos de aprendizagem.

3.6.1 Arquitetura Funcional de Repositório Digital

A arquitetura de repositório digital é resultado da preocupação em padronizar a forma de armazenar documentos em sistemas de aprendizagem. A quantidade de repositórios de documentos e espelhamento dos mesmos tem aumentado, de forma que torna-se necessário facilitar a interoperabilidade entre diferentes repositórios, entre os domínios educacionais e outros.

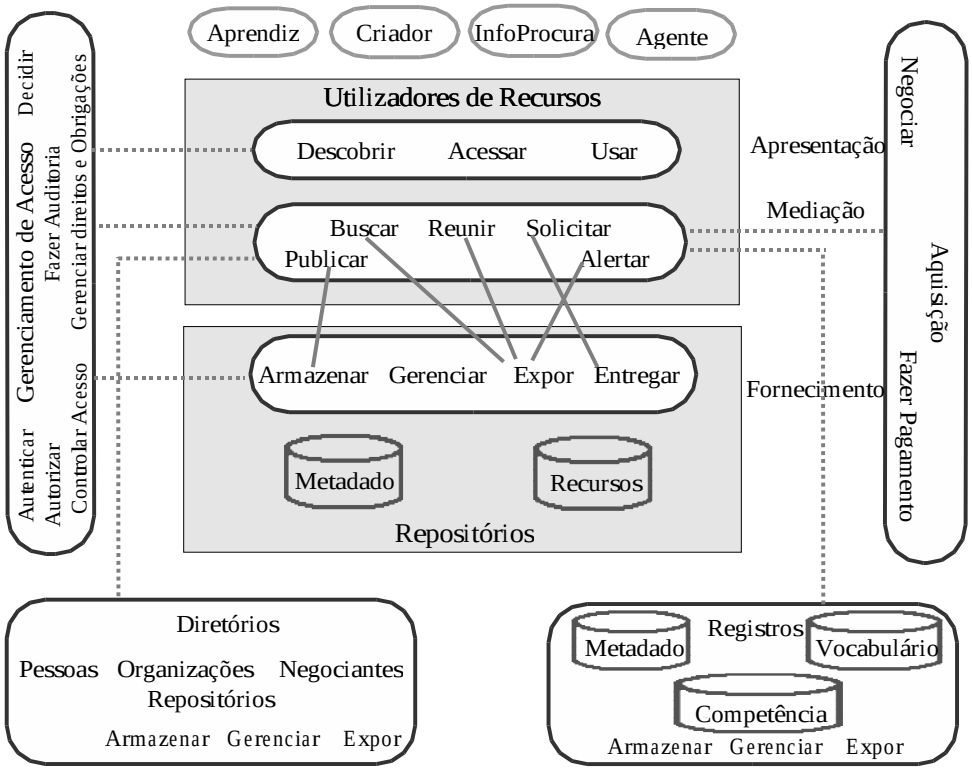

FIGURA 3.4 – Arquitetura Funcional IMS ([44])

Visando facilitar esta interoperabilidade o projeto IMS propõe uma arquitetura funcional de repositório digital, apresentada na Figura 3.4. Esta arquitetura contém três tipos de entidades, define um espaço onde interagem repositórios digitais e serviços de informação. As três entidades são:

- **Usuários**: exemplificado por aprendiz, criador, infoprocura e agente;
- **Componentes funcionais:** para utilizadores de recurso, repositórios, gerenciamento de acesso e aquisição;

- **Serviços:** exemplificado por registros e diretórios.

Os componentes desta arquitetura são:

- **Papéis:** representam um conjunto de funções utilizadas pelos usuários de aplicações educacionais, repositórios digitais e serviços de informação. Os usuários podem receber uma ou mais funções no decorrer de um evento incluindo interações com repositórios.
- **Utilizadores de Recurso:** são aplicações de *software* que permitem aos aprendizes, aos criadores, aos pesquisadores de informação e aos agentes descobrir, acessar e utilizar os recursos e os metadados armazenados em repositórios.
- **Apresentação**: estes componentes funcionais suportam interações entre aprendizes, criadores, pesquisadores de informação, agentes e utilizadores de recurso.
- **Mediação:** constitui componentes funcionais que um utilizador de recurso suporta, permitindo interações com um ou mais repositórios.
- **Repositório:** é uma aplicação de *software* que facilita o armazenamento, o gerenciamento e a entrega de recursos.
- **Recursos:** são os recursos (objetos primários ou secundários, como livros, artigos de jornal, documentos, objetos de aprendizagem, etc) ou metadados (informações descritivas sobre recursos ou outros objetos).
- **Fornecimento**: permite a interação de um repositório com um ou mais utilizadores de recurso.
- **Gerenciamento de Acesso**: existem componentes funcionais que podem estar embutidos dentro de um utilizador de recurso ou repositório de aplicações. Tais componentes realizam o gerenciamento de acesso. Podem ainda ser providos por serviços externos, permitindo o seu compartilhamento entre múltiplos utilizadores de recurso e repositórios.
- **Aquisição**: neste elemento encontramos componentes associados com negociação e pagamento.
- **Diretórios**: este componente representa serviços de diretório externos que suportam as atividades dos utilizadores de recursos e repositórios.
- **Registros**: os serviços de registro externos que suportam as atividades dos utilizadores de recursos e repositórios estão contidos neste elemento da arquitetura.

Os repositórios podem conter recursos atuais ou metadados que descrevem recursos. Os recursos e seus metadados não precisam estar armazenados no mesmo repositório. Os repositórios importantes são aqueles que oferecem recursos para o mundo exterior.

Utilizadores de recurso são quaisquer funções que interagem com os repositórios - estes podem ser um sistema de criação de conteúdo, um LMS, *gateway* de informação, etc.

Os repositórios podem entregar recursos aos utilizadores, que apresentam para o usuário ou para fazer inclusão em outro repositório, como uma cópia do recurso original ou ainda como um componente de um novo recurso.

Dentre os projetos que objetivam implementar arquiteturas de repositórios digitais pode-se destacar o projeto TEAMS, o qual será apresentado na próxima seção.

3.6.2 Artesia - TEAMS

A Artesia envolveu-se na implementação de iniciativas de repositórios digitais para negócios, através de uma faixa de setores de mercados na Europa e nos Estados Unidos. É esperado que estas experiências apresentem um modelo de referência para formular o projeto e a entrada para o grupo de trabalho de repositórios digitais IMS ([44, 7]).

A plataforma TEAMS é um sistema aberto que engloba um paradigma de gerenciamento de recursos digitais, usando tecnologias como Java, SQL, CORBA e XML (incluindo DOM). A Artesia participa ativamente de diversas instituições de padronização, como por exemplo o projeto IMS ([8]).

TEAMS é considerado uma plataforma e um *framework*. Plataforma no sentido de oferecer um repositório digital distribuído, e um *framework* ao oferecer uma aplicação extensível, baseada em componentes, a fim de construir extensões e interfaces de um *site*.

A plataforma TEAMS utiliza os protocolos da família XML para assegurar interoperabilidade entre toda a cadeia de produção e consumo. É possível validar, transformar, pesquisar e montar dados através deste suporte a XML.

A arquitetura é apresentada na Figura 3.5, e contém os seguintes componentes:

- **Interface de Aplicação:** é um aplicativo de navegação pronto para ser utilizado, que é construído com JSP e *servlets* compatíveis com J2EE, e é integrado à lógica de negócio;
- **Serviços de Negócio:** é um *framework* baseado em componentes. Todos os serviços críticos são ligados com CORBA que gerencia o tráfego de rede, eventos de missão crítica e balanceamento de carga.
- **Repositório de Mídia:** consiste em um modelo que pode ser instalado como um repositório tradicional ou como um repositório distribuído. Existem duas partes no repositório de recursos - um armazenamento hierárquico para armazenar os objetos originais, e um banco de dados relacional - para armazenar os metadados.

Outra iniciativa de estender o metadado LOM e definir um modelo de sistema de metadados, é a do projeto europeu *Getting Educational Systems Talking Across Leading-edge Technologies* (GESTALT) que também tem direcionado seus esforços na especificação de uma arquitetura para *e-learning*.

3.7 Projeto GESTALT

O projeto GESTALT estabelece uma ferramenta para o desenvolvimento de sistemas educacionais distribuídos, heterogêneos, escaláveis e compatíveis. O ob-

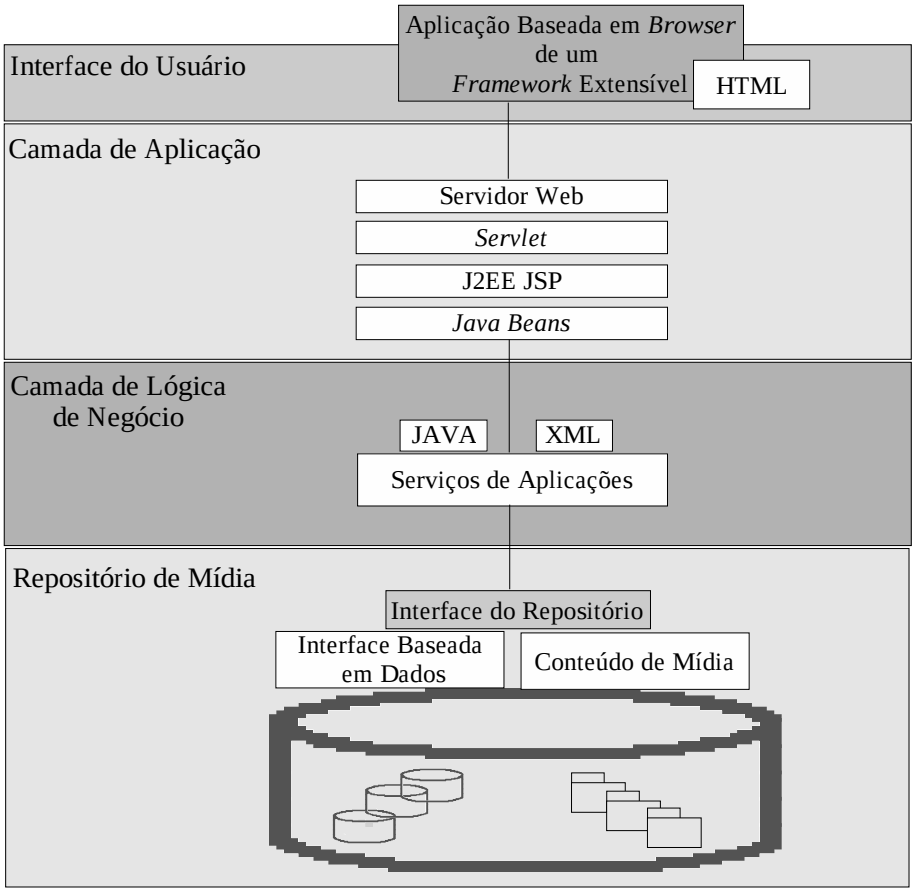

FIGURE 3.5 – Arquitetura Artesia-TEAMS ([8])

jetivo geral desta ferramenta é habilitar usuários a fazer descoberta de recursos educacionais e fornecer acesso à descoberta de recursos através de uma infraestrutura de gerenciamento de rede. Além disso, este projeto contribui com a definição de modelos de dados para sistemas educacionais para rede, especialmente definindo um metadado educacional, bem como perfis e preferências de usuários.

O GESTALT apresenta um metadado chamado *Gestalt Extensions to Metadata STandards for ON-line Educational Systems* (GEMSTONES). Este metadado está baseado no metadado IEEE-LOM e requisitos para descoberta de recursos, ambiente de aprendizagem e *designs* de entrega. No GEMSTONES são observadas algumas extensões em relação ao metadado LOM, que são:

- requisitos técnicos para igualar a capacidade ou configuração de material do cliente;
- tipo de relacionamento adicional para material prévio;
- gênero adicional para cobrir modos de estudo educacional;
- criação de ciclo de vida adicional para provedores de cursos;
- aumento da estrutura de endereço postal;
- formato de dados para incluir dia, mês e ano;

- localização de formato de mídia alternativo;
- gerenciamento de direitos externos.

Além de direcionar seus esforços na especificação do metadados GEMSTONES, o projeto GESTALT apresenta uma arquitetura para sistemas de *e-learning*.

3.7.1 Arquitetura GESTALT

A arquitetura GESTALT para sistemas educacionais apresenta elementos importantes como um serviço de descoberta de recursos, consistindo de um *broker* compatível com CORBA, que é acessado através de um *gateway* Web. Estes serviços permitem aos usuários explorar cursos e módulos e identificar em que instituição está disponível o serviço.

Possui também um sistema de gerenciamento de recursos que tem controle de acesso a recursos, objetos de aprendizagem e capacidade para publicar estes. O gerenciamento de produção de recursos é responsável por coordenar a produção de recursos multimídia e seus metadados relacionados. Apresenta também um servidor de vídeo que suporta múltiplos acessos a serviços multimídia.

Os usuários podem realizar buscas, utilizando descrições de metadados e através destas descrições acessar recursos. As interações com os recursos podem ser realizadas conforme o perfil de cada usuário.

O modelo de arquitetura GESTALT descreve os papéis envolvidos no processo educacional e os relacionamentos entre eles.

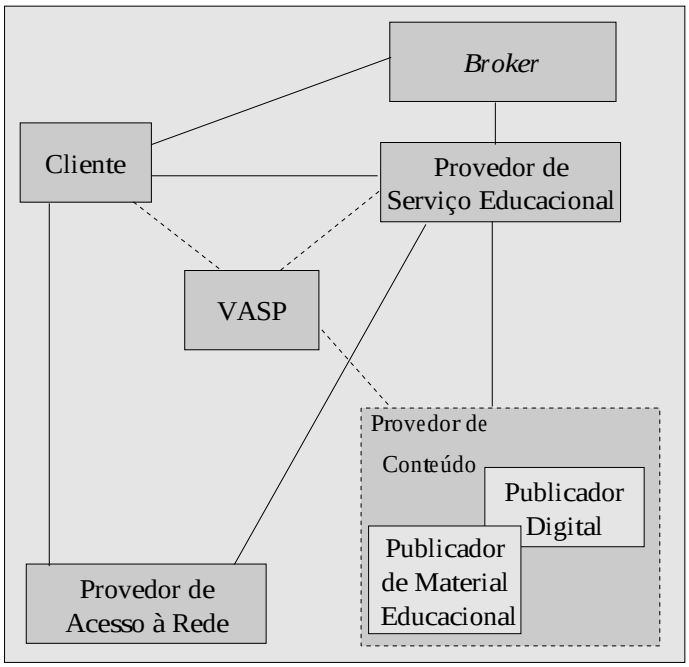

FIGURA 3.6 – O Modelo de Processos e Relacionamentos GESTALT ([28])

3.7.2 Processos Educacionais e Seus Relacionamentos

A Figura 3.6 apresenta os papéis envolvidos nos processos educacionais, os quais são descritos sucintamente a seguir com uma breve descrição dos relacionamentos entre eles.

Cliente

É um indivíduo ou uma organização que procura a existência e a localização de um recurso educacional. O cliente utilizará os serviços de um *broker* para descobrir a existência de um recurso educacional. Com base na especificação fornecida pelo *broker*, o cliente poderá emitir uma requisição de busca com os parâmetros desejados. O *broker* retornará para o cliente as informações sobre o recurso educacional de acordo com os critérios especificados pelo cliente.

O relacionamento entre o cliente e o *broker* está limitado às ações de busca e à localização, não cobrindo aspectos como entrega e envio de recursos educacionais.

A relação existente entre um cliente e um provedor de recursos educacionais é bastante simples. Um cliente irá acessar o servidor educacional para realizar uma sessão. O servidor entregará o conteúdo educacional da requisição.

O cliente também se relaciona com o provedor de acesso à rede, o qual provê as ligações de telecomunicações e serviços necessários para um cliente acessar um servidor, localizado pelo provedor de serviço educacional.

Broker

Pode ser definido como uma entidade on-line que fornece produtos especializados ou informações sobre produtos para clientes que desejam descobrir um produto ou comprar um produto. Produtos são objetos de aprendizagem (componentes ou pacotes dentro de sessões educacionais).

Provedor de Acesso à Rede

É uma organização que oferece as ligações físicas e os serviços de rede.

Provedor de Serviço Educacional

É uma entidade que fornece serviços educacionais para clientes.

O relacionamento entre o *broker* e o provedor de serviços educacionais é basicamente referencial. Ou seja, quando um cliente identifica um recurso educacional de interesse, o *broker* referenciará o cliente para o provedor de serviço educacional apropriado à entrega do recurso.

O provedor de serviço educacional e o provedor de conteúdo atuam no que se refere à provisão de conteúdo digital e material de curso, do provedor de conteúdo para o provedor de serviço.

Provedor de Conteúdo

É um indivíduo ou organização que produz e disponibiliza o conteúdo educacional.

Provedor de Serviço de Valor Agregado (VASP)

É um servidor de perfis de usuários. Um perfil de usuário fornece um ponto comum para o armazenamento consolidado e estruturado de informação sobre um indivíduo e todas as experiências educacionais e vocacionais obtidas pelo indivíduo.

É possível observar que nas arquiteturas apresentadas, alguns componentes se tornam comuns e, desta forma, é possível estabelecer uma arquitetura básica.

3.8 Conclusão

Este capítulo apresentou alguns metadados como o *Dublin Core* e LOM. Também alguns projetos que estão direcionando seus esforços a fim estender o metadado *Dublin Core* para que este possa ser utilizado em sistemas educacionais. O metadado LOM têm sido foco de alguns projetos para sua padronização e extensão.

Os projetos ARIADNE e IMS foram os primeiros a contribuírem na padronização do metadado LOM. Baseado no LOM, o projeto ARIADNE apresenta um sistema próprio de metadados, além de um sistema de repositório de conteúdos educacionais.

O uso de metadados permite a descrição de recursos educacionais. A padronização de metadados permite a interoperabilidade entre sistemas educacionais.

Também foram apresentadas algumas arquiteturas para ambientes educacionais, as quais são resultados de esforços conjuntos de alguns projetos de extensão de metadados e sua utilização.

Foram estudadas arquiteturas propostas por projetos como ARIADNE, IMS e GESTALT. Estas arquiteturas apresentam a utilização de metadados educacionais. Também é possível destacar o esforço por parte do projeto ARIADNE e IMS em especificar uma arquitetura de repositórios digitais, permitindo, assim, a troca de recursos educacionais.

A partir dos estudos realizados, será proposta uma arquitetura que possibilitará a troca de recursos educacionais além de permitir o reuso de serviços educacionais. Esta arquitetura a ser proposta, fará uso de metadados educacionais padrões e da tecnologia de *Web Services* apresentada no Capítulo 2.

Capítulo 4

A Arquitetura Geral da GlueScript

Resumo

Neste capítulo é descrita a proposta de arquitetura que possibilitará a composição de *Web Services* e objetos em formato LOM a partir do uso da linguagem GlueScript.

4.1 Introdução

Nos capítulos anteriores foram apresentados aspectos relacionados à tecnologia de *Web Services* a qual dispõe de um mecanismo de invocação, acesso e reuso de serviços. Este mecanismo apresenta tecnologias padrões como a WSDL, SOAP e UDDI. Estas tecnologias estão baseadas em XML e permitem invocar ou reutilizar um serviço sem a necessidade de conhecer a plataforma ou linguagem de programação usada na sua construção.

A arquitetura de *Web Services* tem características que permitem uma descrição de serviços eficiente através de suas tecnologias padrões.

Através dos metadados educacionais torna-se possível descrever de maneira bastante detalhada qualquer tipo de recurso educacional.

Integrando a tecnologia *Web Services* e os metadados educacionais, teremos uma arquitetura que promoverá a interoperabilidade entre diferentes sistemas educacionais, além de um alto grau de reutilização.

4.2 Bases da Arquitetura de Integração

Nos capítulos anteriores foram estudados aspectos fundamentais para a realização deste trabalho. A arquitetura de *Web Services* apresenta um mecanismo de invocação, acesso e reuso de serviços.

4.2.1 A arquitetura *Web Services*

A arquitetura geral de *Web Services,* no Capítulo 2, apresenta uma estrutura para permitir localização, acesso e reuso de serviços com base em um **Provedor de Serviço** que é a entidade a qual cria o *Web Service* e faz a sua descrição em algum formato padrão e publica os detalhes em um registro central. Um **Solicitante de Serviço** que é uma aplicação que invoca ou inicializa uma interação com um serviço.

Um **Registro de Serviço** onde os provedores de serviços publicam suas descrições dos serviços.

Como mencionado anteriormente, existe a **Descrição do Serviço** que contém os detalhes da interface e de implementação do serviço, incluindo a estrutura de dados, operações e informações de ligação na rede. Também contém dados para facilitar a sua localização pelo **Solicitante de Serviço**.

A arquitetura de *Web Services* utiliza o protocolo SOAP, que é independente de plataforma, de sistema operacional e de linguagem de programação. Além disso, é baseado em XML, tanto nos parâmetros de chamada como nos resultados obtidos. O protocolo SOAP pode utilizar tanto mecanismos de RPC ou de troca de mensagens para acesso aos serviços. O uso mais freqüente de SOAP é sobre o protocolo HTTP, fato este que não exige nenhuma configuração adicional em *firewalls* ou outros dispositivos de rede ([12, 33, 37]).

A arquitetura *Web Services* não está voltada a nenhum domínio específico de aplicações. Observa-se porém que seu uso está intensificado na área de comércio eletrônico. Apresenta características que permitem uma descrição eficiente de serviços através de suas tecnologias padrões.

Além da arquitetura de *Web Services* foi estudado no Capítulo 3 os metadados educacionais e algumas arquiteturas para sistemas educacionais que fazem uso destes. A seção seguinte mostra uma arquitetura genérica com o objetivo de utilizar o metadado LOM.

4.2.2 Arquitetura Genérica de LOM

No Capítulo 3 foram estudados alguns metadados educacionais, os esforços de padronização para que estes possam ser utilizados de forma a permitir a interoperabilidade dos recursos entre diferentes sistemas educacionais. Algumas arquiteturas foram apresentadas neste estudo e foi possível observar o uso de *Common Object Broker Arquitecture* (CORBA) como mecanismo de comunicação e de troca de recursos.

Apesar de permitir a comunicação e a troca de recursos, CORBA utiliza o *Internet Inter-ORB Protocol* (IIOP) e não faz uso de protocolos padrões como por exemplo: HTTP, FTP e SMTP. Por utilizar IIOP como protocolo, a tecnologia CORBA exige que uma nova regra seja criada em *firewalls* corporativos, para que pacotes com IIOP possam trafegar. Por ser um protocolo binário, não é flexível nem dinâmico, sendo necessário recompilar o código devido a quaisquer alterações efetuadas ([37]).

Conforme apresentado no Capítulo 3, através dos metadados educacionais torna-se possível descrever de maneira bastante detalhada qualquer tipo de recurso educacional. Um dos metadados estudados foi o LOM, servindo como base para a definição de uma arquitetura genérica de utilização deste. A Figura 4.1 apresenta a arquitetura composta por um **Cliente** o qual busca por um recurso educacional através de um ***Broker***, este consulta um repositório de metadados educacionais, mais precisamente LOM-XML. O ***Broker*** referencia o **Provedor de Conteúdo** que é responsável por acessar o repositório de recursos educacionais e fazer a entrega do recurso solicitado.

A principal característica da arquitetura LOM é a descrição muito precisa dos recursos educacionais levando em consideração aspectos pedagógicos, o que não é possível através da descrição dos serviços através da WSDL, utilizada na arquitetura

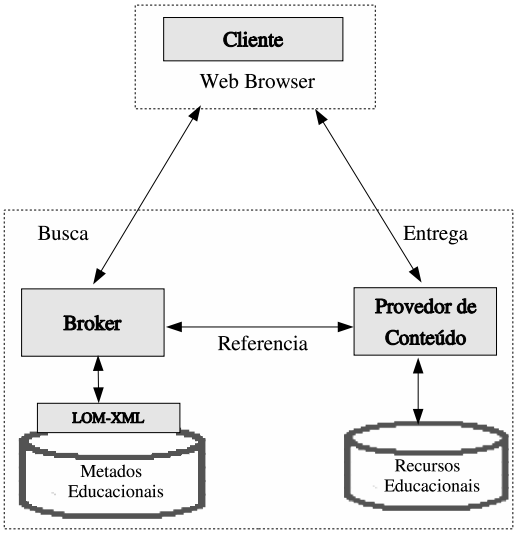

FIGURA 4.1 – Arquitetura Genérica LOM

de *Web Services*.

Portanto a tecnologia de *Web Services* é bem definida para permitir busca e acesso de serviços de forma bastante eficiente através da descrição WSDL, registro UDDI e do protocolo SOAP. Por outro lado, temos a arquitetura de LOM que apresenta uma descrição de recursos educacionais de forma detalhada.

A seção seguinte apresenta uma arquitetura genérica que faz uso das duas tenologias mencionadas anteriormente.

Integrando a tecnologia *Web Services* e os metadados educacionais é possível criar uma arquitetura para promover a interoperabilidade entre diferentes sistemas educacionais, além de um alto grau de reutilização. A interoperabilidade, composição e reuso em ponto grande será atingida pela definição da linguagem GlueScript.

A interoperabilidade está em permitir que recursos e serviços possam de alguma forma, interagir entre si. Já a composição torna-se possível a partir do momento em que se possa construir uma nova aplicação utilizando serviços e recursos já existentes. Além disso, a composição também pode ser realizada quando um serviço ou recurso utiliza como entrada os dados de saída de outro. Desta forma, o fato de utilizar recursos e serviços já existentes implica em alto grau de reuso.

4.3 Arquitetura da *Domain Specific Language* (DSL) GlueScript

A DSL utiliza a arquitetura da Figura 4.2 para a realização de suas operações.

Esta arquitetura é composta pela estrutura padrão de *Web Services* e uma arquitetura básica utilizada com metadados educacionais. A arquitetura proposta apresenta uma **Camada de Composição de Serviços**, a qual permite que seja feita busca de serviços utilizando-se a estrutura de *Web Services* existente, bem como fazer uso de recursos educacionais especificados por LOM-XML ([43]).

Esta nova camada permite que serviços sejam encontrados tanto em ambientes *Web Services* como em ambientes específicos de educação. Será possível manipular

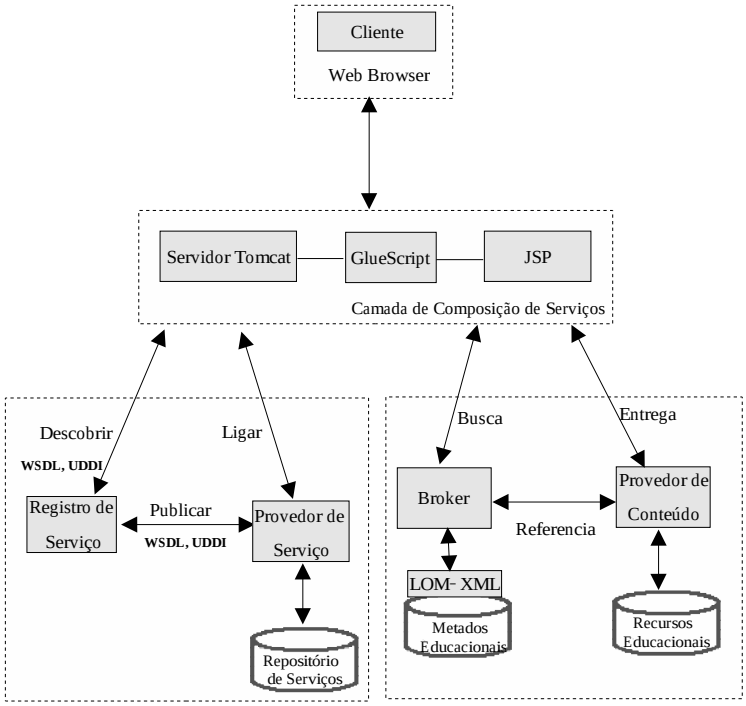

FIGURA 4.2 – Arquitetura da GlueScript

estes serviços para a construção de novas aplicações para ambientes educacionais de forma dinâmica. As formas de acesso a *Web Services* e recursos educacionais disponíveis até o momento permitem apenas o acesso estático. Com a utilização desta nova camada, a localização e acesso tornam-se dinâmicos o que confere caráter diferenciado a arquitetura desenvolvida.

4.3.1 Camada de Composição

A camada de composição é composta por um servidor Web Tomcat, pela linguagem JSP e pela GlueScript.

Características de algumas linguagens scripts foram observadas com o objetivo de encontrar uma para que a GlueScript pudesse estendê-la. Dentre elas, podemos relacionar as que seguem:

- ASP: roda no lado servidor, criada pela Microsoft, permite utililizar diversas linguagens de *script* como Visual Basic e JScript. Roda em servidores *Internet Information Service* (IIS) ou *Personal Web Service* (PWS). Assim sendo, ASP está ligada a plataforma Windows.

- PHP: roda no lado servidor. PHP é *open source* e multiplataforma. Roda em servidores IIS e Apache.

- Perl: baseada em *Common Gateway Interface* (CGI). Perl é open source e multiplataforma. Os servidores HTTP (IIS e Apache), invocam o intepretador Perl.

- JSP: roda no lado servidor, foi criada pela Sun é multiplataforma e roda em servidores como Tomcat, BEA WebLogic, IBM WebSphere e outros que

tenham *Java Virtual Machine* (JVM). Os códigos JSP são transformados em *servlets* em *background*.

A linguagem JSP é baseada em Java o que a torna portável. É utilizada através de *scriptlets* e *tags*. Os *scriptlets* são escritos em Java e exigem do programador conhecimento da sintaxe Java. As *tags* apresentam uma sintaxe simples e permitem fácil integração no ambiente de desenvolvimento.

É possível implementar um conjunto de *tags* que são chamadas de *Tag Libraries*. Estas JSP *Tag Libraries* permitem estender JSP. Através das *Tag Libraries* torna-se possível separar a camada de apresentação da camada de implementação, conforme a Figura 4.3. Assim, o programador não necessita ter conhecimento da linguagem de programação Java para implentar suas aplicações.

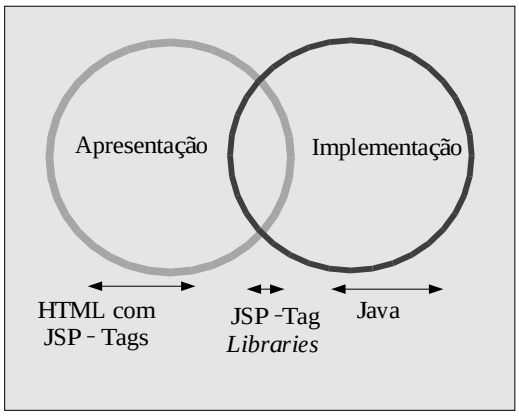

FIGURA 4.3 – Separação da Camada de Apresentação da Implementação ([60])

A linguagem de composição GlueScript é uma extensão de JSP. Será utilizada através de uma *Tag Library*. Desta forma, os desenvolvedores de ambientes para educação a distância poderão utilizar a GlueScript sem necessitar conhecer programação Java.

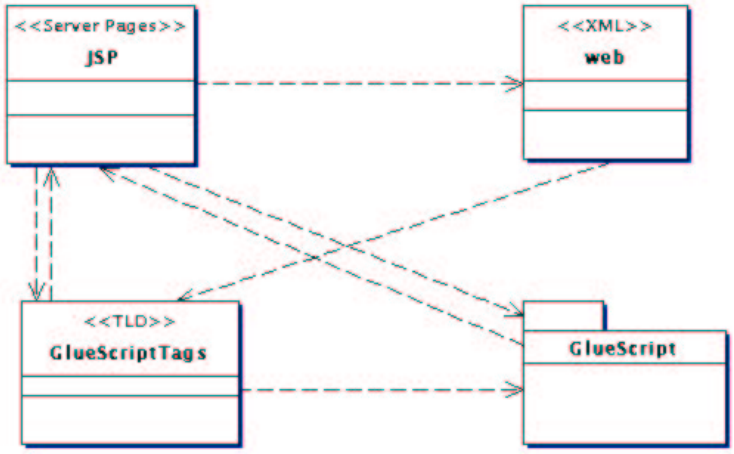

FIGURA 4.4 – Diagrama UML da Arquitetura Geral da GlueScript

A Figura 4.4 apresenta um diagrama UML idealizando o uso da *Tag Library* da GlueScript. O desenvolvedor Web poderá fazer uso dos recursos já disponíveis em JSP e fazer uso das *tags* da GlueScript. Quando o servidor for compilar os códigos JSP, e encontrar *tags* que não forem padrão do JSP, o servidor através de um arquivo web.xml localizará o arquivo que armazena as *tags* da GlueScript. Um arquivo GlueScriptTags referenciará as *tags* e os parâmetros possíveis de serem utilizados.

A GlueScript permite, por seus comandos algumas funcionalidades como localizar serviços ou recursos educacionais, através de documentos WSDL e do metadado LOM-XML. Também que estes recursos possam ser acessados e reutilizados em forma de composição à criação de ambientes educacionais para Web.

O protocolo SOAP é utilizado para localização e acesso de recursos e serviços. O uso de SOAP está na facilidade de uso, o que permite que os serviços localizados possam ser acessados e reutilizados sozinhos ou em conjunto com outros serviços.

No Capítulo 1 foi apresentado um conjunto de APIs Java para utilização dos padrões SOAP, WSDL e UDDI. A GlueScript é implemetada em Java e desta forma faz uso do pacote JWSDP da Sun para manipulação dos serviços e recursos educacionais.

Os comandos necessários para localização, acesso e reuso de serviços são disponibilizados pela linguagem GlueScript.

O próximo capítulo apresenta detalhes da linguagem de composição GlueScript bem como sua estrutura e descrição dos seus comandos.

4.4 Conclusão

Esta arquitetura permite que serviços sejam encontrados tanto em ambientes *Web Services* como em ambientes específicos de educação. Será possível manipular estes serviços para a construção de novas aplicações para ambientes educacionais.

Isso torna-se possível através da camada de composição que permite a localização e acesso de serviços que são descritos e disponibilizados através da arquitetura de *Web Services* bem como de recursos educacionais descritos através do metadado LOM.

Capítulo 5

GlueScript: Uma Linguagem Específica de Domínio para Composição de *Web Services*

Resumo

Este capítulo apresenta a DSL GlueScript. Esta linguagem de *script* permite que utilizando-se a arquitetura proposta no Capítulo 4, objetos educacionais e/ou *Web Services* possam ser compostos para construção de ambientes educacionais.

5.1 Introdução

Uma linguagem de composição deve permitir a colaboração e o reuso de serviços já existentes. Desta forma, este capítulo apresenta algumas linguagens de domínio específico como por exemplo Piccola, WebL, Mawl e WebCompose.

O capítulo anterior apresentou uma arquitetura genérica para composição de Web Services e recursos educacionais. A arquitetura proposta serve de base para a definição da GlueScript a fim de promover os objetivos de composição, reuso e interoperabilidade.

Esta linguagem é uma extensão de *Java Server Pages* (JSP), sendo utilizada através de uma *Tag Library* que será especificada neste capítulo.

5.2 Linguagens de Composição

As linguagens de composição apresentam características as quais permitem a colaboração entre serviços. Além disso, permitem que a partir de uma coleção de componentes utilizáveis programados em outras linguagens, seja possível combiná-los, bem como estender algumas de suas características.

5.2.1 Piccola

Piccola foi desenvolvida por Franz Achermann, Markus Lumpe, Jean-Guy Schneider e Oscar Nierstrasz ([1]).

Piccola é uma linguagem de composição baseada em objetos e está baseada na teoria de $\pi\mathcal{L}$-*calculus*. Utiliza o conceito de programação funcional. O principal elemento da linguagem é uma expressão de ordem que representa o conceito unificado de $\pi\mathcal{L}$-*agents* e $\pi\mathcal{L}$-*forms*. Expressões de ordem são seqüências de termos de ordem (ex: chamadas de funções síncronas e assíncronas). Possui uma sintaxe semalhante à *Python* e *Hashell*.

Piccola possui um protótipo feito em Java e um Java *Gateway* para interconexões com componentes externos. Desta forma, é possível encapsular objetos java dentro de *scripts* Piccola.

O *Gateway* Piccola atua em conjunto com o *servlet* WWW, permitindo interpretar todas as requisições e assim oferecer mecanismos de composição.

A linguagem Piccola utiliza o conceito de *Blackboard* para um servidor WWW, onde a cada requisição de um *Blackboard* é criada uma instância particular para o módulo que o requisitou.

5.2.2 WebL

A WebL foi projetada e implementada por Thomas Kistler e Hannes Marais ([54]). A contribuição de Luca Cardelli e Rowan Davies para a linguagem foram os combinadores de serviço([16]). Tom Rodeheffer sugeriu melhorias à implemetação assim como Monika Henzinger, Jeff Dean, Brian Eberman e Jin Yu contribuíram na melhoraria da linguagem e eliminar *bugs*. Atualmente a WebL é a linguagem Web da Compaq ([54]).

A WebL é uma linguagem *script* para processamento de documentos. Utilizada com o objetivo de recuperar documentos da Web, extrair informações e manipular conteúdos de documentos.

Uma característica importante da linguagem é a manipulação de textos em formato como HTML e XML. A WebL também suporta características que simplificam a manipulação de falhas de comunicação, a exploração de documentos replicados sobre múltiplos serviços Web, para confiabilidade e realização de múltiplas tarefas em paralelo. WebL também fornece características de linguagens de programação imperativas tradicionais como objetos, módulos, estruturas de controles, etc.

O modelo de computação da WebL está baseado nos conceitos chamados de combinação de serviços e álgebra de marcação. Combinação de serviços é um formalismo que pode prover acesso mais confiável a recursos e serviços Web. Álgebra de marcação é um formalismo para extrair informação de textos estruturados e manipulação destes documentos.

A WebL foi implementada em Java. Os objetos Java podem ser chamados diretamente da WebL sem a necessidade de extensão.

5.2.3 Mawl

A linguagem Mawl foi desenvolvida em 1995 ([9]), em virtude das dificuldades dos programadores que utilizavam programas CGI e o protocolo HTTP. Mawl apresenta uma arquitetura de serviços baseada em formulários, que é independente do protocolo HTTP e CGIs.

Um serviço em Mawl consiste de uma ou mais sessões. Uma sessão estabelece o fluxo de controle de um serviço e permite a atualização de variáveis. Ou seja, uma sessão é um programa seqüencial que se comunica com o usuário fazendo chamadas

de métodos em formulários. Mawl apresenta o conceito de *template* que é uma porção estática ou dinâmica de uma interface do usuário que pode ser parametrizada para exibir informações personalizadas. Cada formulário tem um ou mais *templates* associados. O *template* é por exemplo, um documento em formato HTML.

Mawl possui uma linguagem chamada MHTML que é uma extensão de HTML e permite adicionar código Mawl dentro de páginas em formato HTML. As *tags* adicionais em MHTML permitem a substituição de dados de serviços em um formulário.

A composição de serviços em Mawl acontece no momento em que MHTML é utilizado para especificar comportamentos específicos dentro de páginas HTML.

5.2.4 WebCompose

A WebCompose foi desenvolvida por Sérgio Crespo C. da S. Pinto ([57]). É uma linguagem específica de domínio (DSL) que utiliza uma abordagem de *framelets* e é baseada na teoria de ADVService e ADOService.

Através do paradigma de *framelets,* cada comando da linguagem comporta-se como um pequeno *framework*, com seus pontos de flexibilização definidos e com suas formas de utilização que são *black-box* e *white-box.*

Os comandos da linguagem oferecem uma série de recursos à criação de restrições, manipulação de formulários, banco de dados, utilização de *pipes and filters* e serviços de *proxy* entre os serviços.

A WebCompose tem como objetivo a utilização de serviços já existentes na Web e que não foram necessariamente projetados para atuarem em conjunto ou cooperarem com outros serviços. Foi desenvolvida para ser utilizada em conjunto com CGILua e Lua, permitindo uma extensão da ferramenta CGILua. Desta forma, ela pode transformar um servidor CGI que utiliza a ferramenta CGILua a fim de processar *scripts* de páginas Web em um servidor ADOService WWW que admite a composição e reutilização de serviços presentes na Web.

Outra característica importante é a disponibilização de um *Blackboard* no servidor ADOService WWW, que autoriza que dados sejam compartilhados por diversos serviços de forma assíncrona.

As linguagens de composição mencionadas anteriormente apresentam características bastante interessantes quanto a maneira de manipular dados de formulários ou compor serviços de alguma forma.

A criação de uma linguagem para composição de *Web Services* e recursos educacionais tem como principal objetivo permitir a construção de ambientes educacionais utilizando as facilidades disponíveis da tecnologia de *Web Services* e LOM-XML.

A GlueScript mostra características que a diferem das linguagens mencionadas nesta seção. Estas características serão mostradas de forma detalhada nas seções que seguem.

5.3 GlueScript

A linguagem GlueScript é uma extensão de *JavaServer Pages* (JSP), para permitir a composição de *Web Services* e de recursos educacionais. O diferencial de utilizar JSP está em ser uma linguagem *server side,* diferente das linguagens que utilizam programação CGI ([62]).

Na programação com CGI, o servidor Web executa os programas externos e

passa as informações de requisições HTTP, para o programa que o requisitou. A resposta do programa externo é então passado de volta para o servidor Web o qual disponibiliza-o ao *browser* cliente. Para N requisições, o programa CGI é carregado N vezes na memória. Isto pode acarretar um alto consumo de ciclos de CPU e sobrecarga no uso de memória ([36, 49]).

As páginas JSP quando compiladas são transformadas em *servlets*. Com *servlets*, a máquina virtual Java fica rodando e manipula cada requisição utilizando *threads* Java leves e não processos pesados do sistema operacional. Ao contrário de CGIs, os *servlets* podem ter N *threads* mas somente uma cópia da classe *servlet* na memória ([62, 49]).

Os comandos da GlueScript foram implementados em Java e são utilizados através de uma *Tag Library* JSP ([60]). O pacote *Java Web Services Development Pack* (JWSDP), que apresenta um conjunto de APIs Java para XML é utilizado a fim de facilitar o uso do protocolo SOAP e o acesso aos registros UDDI, WSDL e LOM-XML. A principal característica no uso dos comandos da GlueScript está em localizar e acessar de forma dinâmica os *Web Services* e recursos educacionais.

<GLUESCRIPT>	::= <TAG>
<TAG>	::= <TAG> <TAG> | '<'<TAG>'>' '</'<TAG>'>' |
	<LOCATEWS> | <LOCATELOM> | <PARSINGWS> |
	<PARSINGLOM> | <ALLOCATEWS> |
	<ALLOCATELOM> | <PIPE> | Λ
<LOCATEWS>	::= LOCATEWS <PARAM> | LOCATEWS
<LOCATELOM>	::= LOCATELOM <PARAM> | LOCATELOM
<PARSINGWS>	::= PARSINGWS <PARAM> | PARSINGWS
<PARSINGLOM>	::= PARSINGLOM <PARAM> | PARSINGLOM
<ALLOCATEWS>	::= ALLOCATEWS <PARAM> | ALLOCATEWS
<ALLOCATELOM>	::= ALLOCATELOM <PARAM> | ALLOCATELOM
<PIPE>	::=PIPE<PARAM> | PIPE
<PARAM>	::= <PARAM> <PARAM> | <URL> | <STRING>

FIGURA 5.1 – BNF da GlueScript

A Figura 5.1 apresenta a estrutura básica das *tags* em *Backus-Naur Form* (BNF). Na seção que segue são apresentadas as funcionalidades destas *tags*.

5.3.1 Funcionalidades das Tags

As *tags* foram implementadas em Java e para sua utilização em conjunto com JSP, torna-se necessário o uso de uma estrutura específica com a finalidade de que o servidor de páginas possa compilar e reconhecer as novas *tags* desenvolvidas.

Esta estrutura é composta por um documento web.xml que é consultado pelo servidor (Tomcat) quando um documento JSP é compilado. Esse documento é necessário pois as novas *tags* desenvolvidas precisam ser reconhecidas na compilação. O documento web.xml contém a localização do documento GlueScriptTags.tld.

O documento GlueScriptTags.tld apresenta todas as *tags* que foram desenvolvidas, juntamente com os seus parâmetros. Esse arquivo é o vínculo entre o arquivo

JSP criado pelo desenvolvedor e a parte funcional da linguagem implementada em Java.

FIGURA 5.2 – Localização dos Arquivos da GlueScript

Estes arquivos são dispostos em um diretório do servidor Tomcat e seguem a estrutura conforme a Figura 5.2.

Web.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">

<web-app>
 <taglib>

  <taglib-uri>                                  1
   http://localhost:8080/glues
  </taglib-uri>

  <taglib-location>                             2
   /WEB-INF/GlueScriptTags.tld
  </taglib-location>

 </taglib>
</web-app>
```

GlueScriptTags.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
  PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
  "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>
 <tlibversion>1.0</tlibversion>
 <jspversion>1.2</jspversion>
 <shortname>glues</shortname>
 <uri> http://localhost:8080/glues </uri>
 <info>
  DSL GlueScript Versão 0.1
 </info>

 <tag>                                          3
  <name>GlueScript</name>
  <tagclass>GlueScript.GlueScript</tagclass>
  <bodycontent>JSP</bodycontent>
  <info>
   Tag principal, contém as demais.
  </info>
 </tag>
```

FIGURA 5.3 – Relacionamento dos Arquivos web.xml e GlueScriptTags.tld

O relacionamento entre os arquivos Web.xml e GlueScriptTags.tld é apresentado na Figura 5.3 como:

1. Define o endereço da localização de todos os arquivos relacionados para que possam ser encontrados pelo Tomcat.

2. Define a localização do arquivo descritor das *tags* (GlueScriptTags.tld). Desta forma, o servidor pode localizar o descritor de *tags*, o qual apresenta a relação de todas as *tags* desenvolvidas, os parâmetros (quando houver) e o nome da classe Java relacionada.

3. Apresenta a definição de uma *tag* (GlueScript) e sua classe Java correspondente no pacote GlueScript. Esta é a *tag* principal do conjunto de *tags*. Em virtude do arquivo descritor de *tags* ser de um tamanho considerável, ele será exposto, em partes, conforme as demais *tags* forem sendo apresentadas.

5.3.2 *Tag* GlueScript

A *tag* GlueScript é necessária para a utilização de todas as outras *tags* desenvolvidas. Esta *tag* é responsável por armazenar os parâmetros que são retornados pelas demais *tags* da linguagem.

A Figura 5.4 mostra um exemplo de uso da *tag* GlueScript em um arquivo JSP. As demais *tags* desenvolvidas são utilizadas dentro do escopo da *tag* principal. As *tags* são dispostas de forma aninhada sendo que a *tag* que estiver no nível acima recebe os dados de retorno. Assim sendo, os dados armazenados pelas *tags* podem ser manipulados através de JSP.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<html>
 <title>Teste da GlueScript</title>
 <body>
   <gs:GlueScript>
     //Outras Tags
   </gs:GlueScript>
 </body>
</html>
```

FIGURA 5.4 – A Tag GlueScript

5.3.3 *Tags* LocateWS e LocateLOM

Estas *tags* fazem a localização de *Web Services* e recursos educacionais em registros. Elas são utilizadas quando o desenvolvedor não conhece a localização específica de um *Web Service* ou de um recurso educacional. Através do endereço de um registro UDDI ou de um registro de metadados LOM é possível saber onde estão os documentos de descrição de *Web Services* e recursos educacionais.

5.3.3.1 LocateWS

A *tag* LocateWS é utilizada para localização de *Web Services*. Através da API Java para registros XML (JAXR), é possível acessar registros de negócios e informações relacionadas. Os registros UDDI são acessados e então é possível saber a localização de um documento WSDL que descreve o *Web Services* definindo a sua localização, seus métodos e a forma de acesso.

GlueScriptTags.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
  PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
  "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>
 <tlibversion>1.0</tlibversion>
 <jspversion>1.2</jspversion>
 <shortname>glues</shortname>
 <uri> http://localhost:8080/glues </uri>
 <info>
  DSL GlueScript Versão 0.1
 </info>
<tag>
  <name>LocateWS</name>
  <tagclass>GlueScript.LocateWS</tagclass>          1
  <bodycontent>JSP</bodycontent>
  <info>
   Localiza WSs no registro passado como parâmetro.
  </info>
  <attribute>
   <name>address</name>                             2
   <required>true</required>                        3
   <rtexprvalue>true</rtexprvalue>                  4
  </attribute>
 </tag>
```

FIGURA 5.5 – GlueScriptTags - Descrição da *tag* LocateWS

A Figura 5.5 apresenta o arquivo GlueScritptTags com a *tag* LocateWS onde são definidos:

1. O nome da *tag* e a classe Java correspondente no pacote GlueScript.
2. O parâmetro que deve ser passado para a *tag*. O atributo *address* indica o endereço de um registro UDDI.
3. Define se o atributo é obrigatório ou não.
4. Permite a inserção de *scriptlets* JSP, ou seja, o parâmetro *address* pode ser passado para a *tag* através de um *scriptlet* JSP.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<html>
 <title>Teste da GlueScript</title>
 <body>
   <gs:GlueScript>
     <gs:LocateWS address="http://localhost:8089/registry-server/RegistryServerServlet"/>
   </gs:GlueScript>
 </body>
</html>
```

FIGURA 5.6 – Utilizando a *Tag* LocateWS

A Figura 5.6 apresenta um exemplo de uso da *tag* LocateWS. Ao utilizar esta *tag*, um registro UDDI é acessado com o objetivo de encontrar os *Web Services*. O

atributo *address* é um endereço de um registro UDDI. Esta *tag* retorna o nome de todos os serviços contidos em um registro UDDI, bem como a URL do documento WSDL de cada um conforme a Figura 5.7. É importante salientar que a *tag* LocateWS foi utilizada no exemplo em conjunto com a *tag* GlueScript, e neste caso, o resultado é retornado para esta *tag,* como já mencionado anteriormente. Assim sendo, o resultado mostrando todos os documentos WSDLs contidos no registro UDDI podem ser manipulados pelo desenvolvedor usando a própria GlueScript ou através de JSP.

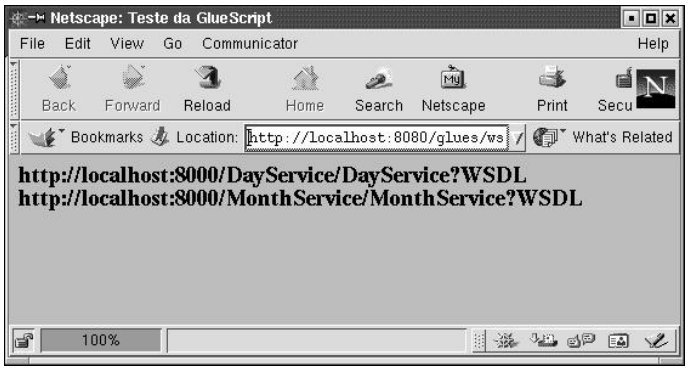

FIGURA 5.7 – Resultado da *Tag* LocateWS

5.3.3.2 LocateLOM

A *tag* LocateLOM é utilizada para localização de recursos educacionais descritos através de LOM-XML. Os registros LOM-XML são acessados com a finalidade de encontrar os recursos disponíveis através destes registros. Foi utilizada a API Java para Mensagens XML (JAXM). Através desta API é possível enviar e receber mensagens SOAP pela Internet de forma eficiente e transparente. Assim mensagens SOAP são enviadas a um registro de metadados educacionais com a finalidade de descobrir a localização de um documento LOM-XML. A mensagem de retorno indica onde este documento está localizado, e, a partir deste documento é possível obter as informações necessárias sobre um determinado recurso educacional.

GlueScriptTags.tld

```
<tag>
  <name>LocateLOM</name>                                    1
  <tagclass>GlueScript.LocateLOM</tagclass>
  <bodycontent>JSP</bodycontent>
  <info>
    Localiza metadados educacionais no registro passado como parâmetro.
  </info>
  <attribute>
    <name>address</name>                                    2
    <required>true</required>                               3
    <rtexprvalue>true</rtexprvalue>                         4
  </attribute>
</tag>
```

FIGURA 5.8 – GlueScriptTags - Descrição da *tag* LocateLOM

A Figura 5.8 apresenta parte do documento descritor de *tags*, o qual descreve a *tag* LocateLOM:

1. O nome da *tag* e a classe Java correspondente no pacote GlueScript.
2. O parâmetro que deve ser passado para a *tag*. O atributo *address* indica o endereço de um registro de metadados LOM.
3. Define se o atributo é obrigatório ou não.
4. Permite a inserção de *scriptlets* JSP ou seja, o parâmetro *address* pode ser passado para a *tag* através de um *scriptlet* JSP.

Um exemplo de uso da *tag* LocateLOM é exposto na Figura 5.9.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<html>
 <title>Teste da GlueScript</title>
 <body>
  <gs:GlueScript>
   <gs:LocateLOM address="http://localhost:8080/lom/metadata.index"/>
  </gs:GlueScript>
 </body>
</html>
```

FIGURA 5.9 – Utilizando a *Tag* LocateLOM

5.3.4 *Tags* ParsingWS e ParsingLOM

Quando os registros são acessados através das *tags* LocateWS e LocateLOM, são localizados onde estão os documentos de descrição destes serviços ou recursos (WSDL). Por conseguinte é preciso através destes documentos conhecer detalhes como a localização destes serviços, bem como todas as informações necessárias para que estes possam ser acessados. As *tags* ParsingWS e ParsingLOM, permitem analizar os documentos de descrição de *Web Services* e recursos educacionais para obtenção de informações como a localização, o tipo de recurso ou serviço dentre outras. Sempre que as *tags* LocateWS e LocateLOM forem utilizadas, torna-se necessário o uso das *tags* ParsingWS e ParsingLOM.

5.3.4.1 ParsingWS

Após a localização de um documento WSDL, a utilização da API Java para processamento XML (JAXP) permite que se faça processamento de documentos XML utilizando vários *parsers* como *Simple API for XML parsing* (SAX) e *Document Object Model* (DOM).

Utilizar um *parser* nos documentos WSDL é necessário para que o desenvolvedor saiba onde localizar o *Web Services* e quais métodos são necessários a fim de que o mesmo possa ser invocado.

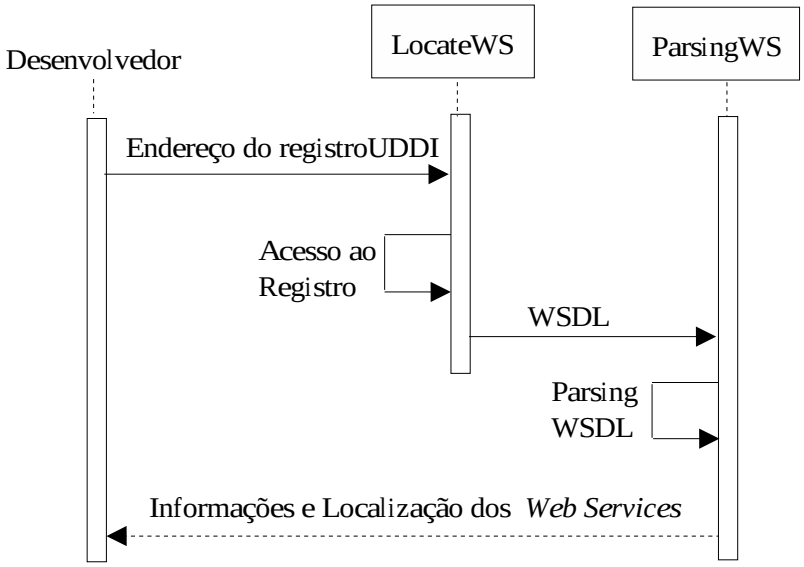

FIGURA 5.10 – Diagrama de Seqüência das *tags* LocateWS e ParsingWS

O diagrama de seqüência apresentado na Figura 5.13 mostra o relacionamento entre as tags LocateWS e ParsingWS.

GlueScriptTags.tld

```
<tag>
  <name>ParsingWS</name>          1
  <tagclass>GlueScript.ParsingWS</tagclass>      2
  <bodycontent>JSP</bodycontent>
  <info>
    Efetua o parsing de um WSDL passado como parâmetro.
  </info>
</tag>
```

FIGURA 5.11 – GlueScriptTags - Descrição da *tag* ParsingWS

A Figura 5.11 apresenta a *tag* ParsingWS definida no arquivo GlueScript.tld:

1. Define o nome da *tag* a ser utilizada dentro de uma página JSP.
2. Define a classe Java acessada pela *tag* no pacote da GlueScript.

Na Figura 5.12 é apresentado um exemplo de uso da *tag* ParsingWS. O endereço do documento WSDL retornado pela *tag* LocateWS é utilizado para realizar o *parsing* do documento e extrair os métodos que o desenvolvedor poderá utilizar para a invocação do *Web Service*.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<html>
 <title>Teste da GlueScript</title>
 <body>
  <gs:GlueScript>
    <gs:ParsingWS>
      <gs:LocateWS address="<%=registry%>"/>
    </gs:ParsingWS>
  </gs:GlueScript>
 </body>
</html>
```

FIGURA 5.12 – Utilizando a *Tag* ParsingWS

5.3.4.2 ParsingLOM

Após um arquivo LOM-XML ser localizado é preciso realizar o *parsing* deste documento para que o desenvolvedor possa obter as informações necessárias para acessar o recurso. Além disso é possível obter informações pedagógicas pertinentes ao recurso, caso seja de interesse do desenvolvedor. Como mencionado no Capítulo 3, um arquivo LOM-XML pode conter informações detalhadas de um recurso. Estas informações podem ser o nível de aprendizado do recurso, a quem se destina, se há algum custo, outros recursos relacionados, autor, instruções de uso dentre outras informações.

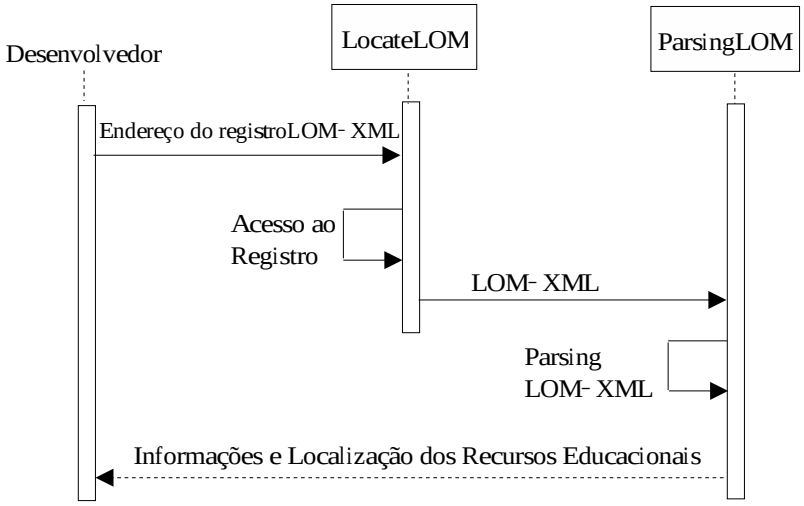

FIGURA 5.13 – Diagrama de Seqüência das *tags* LocateLOM e ParsingLOM

O diagrama de seqüência apresentado na Figura 5.13 mostra o relacionamento entre as tags LocateLOM e ParsingLOM.

GlueScriptTags.tld

```
<tag>
  <name>ParsingLOM</name> 1
  <tagclass>GlueScript.ParsingLOM</tagclass> 2
  <bodycontent>JSP</bodycontent>
  <info>
    Efetua o parsing de um documento LOM- XML passado como parâmetro.
  </info>
</tag>
```

FIGURA 5.14 – GlueScriptTags - Descrição da *tag* ParsingLOM

A Figura 5.14 apresenta a definição da *tag* ParsingLOM. A *tag* é definida da seguinte forma:

1. Nome da *tag* para ser utilizado em páginas JSP.
2. Nome da classe Java correspondente no pacote GlueScript.

Um exemplo de uso da *tag* ParsingLOM é exposto na Figura 5.15.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<html>
 <title>Teste da GlueScript</title>
 <body>
   <gs:GlueScript>
     <gs:ParsingLOM>
       <gs:LocateLOM address="http://localhost:8080/lom/metadata.index"/>
     </gs:ParsingLOM>
  </gs:GlueScript>
 </body>
</html>
```

FIGURA 5.15 – Utilizando a *Tag* ParsingLOM

5.3.5 *Tags* AllocateWS e AllocateLOM

Estas *tags* permitem a utilização de *Web Services* e recursos educacionais. Um desenvolvedor pode utilizar estas *tags* após ter obtido as informações de localização de um recurso ou serviço, assim como quando as informações referentes a localização e formas de acessso de um recurso ou serviço já são conhecidas. Assim um *Web Services* ou um recurso educacional pode ser acessado ou ser composto no desenvolvimento das aplicações Web.

5.3.5.1 AllocateWS

Esta *tag* invoca um *Web Service* utilizando a localização e os métodos necessários passados como parâmetros. A implementação da *tag* em Java utiliza a API Java para XML baseada em RPC (JAX-RPC) que torna possível escrever uma aplicação em linguagem de programação Java, que utiliza SOAP para fazer uma *Remote Procedure Call* (RPC). Também pode servir no envio de mensagens de requisição e resposta.

A Figura 5.16 apresenta a definição da *tag* AllocateWS no arquivo GlueScriptTags.tld. O arquivo especifica a *tag* e os atributos necessários para que um *Web Service* possa ser invocado. Os atributos descritos são originados do *parsing* realizado no documento WSDL. Estes atributos são especificados como segue:

1. Define o nome da *tag* e a classe Java correspondente no pacote.

GlueScriptTags.tld

```
<tag>
 <name>AllocateWS</name>
 <tagclass>GlueScript.AllocateWS</tagclass>                    1
 <bodycontent>JSP</bodycontent>
 <info>
  Utiliza um método de um WS.
 </info>
 <attribute>
  <name>endpoint</name>
  <required>true</required>                                    2
 </attribute>
 <attribute>
  <name>service</name>
  <required>true</required>                                    3
 </attribute>
 <attribute>
  <name>portn</name>
  <required>true</required>                                    4
 </attribute>
 <attribute>
  <name>tns</name>
  <required>true</required>                                    5
 </attribute>
 <attribute>
  <name>method</name>
  <required>true</required>                                    6
 </attribute>
 <attribute>
  <name>parameters</name>
  <required>true</required>                                    7
 </attribute>
 <attribute>
  <name>valores</name>
  <required>true</required>                                    8
 </attribute>
 <attribute>
  <name>pipe</name>
  <required>true</required>                                    9
 </attribute>
</tag>
```

FIGURA 5.16 – GlueScriptTags - Descrição da *tag* AllocateWS

2. Atributo *endpoint*: é o endereço para acesso do *Web Service*.
3. Atributo *service*: é o nome do *Web Service* que será invocado.
4. Atributo *portn*: é a porta para a qual será enviado o pedido do serviço.
5. Atributo *tns*: é o *namespace* do *Web Service*.
6. Atributo *method*: é o nome do método que será invocado.
7. Atributo *parameters*: são os parâmetros que serão passados para o método.
8. Atributo *valores*: são os dados dos parâmetros especificados.
9. Atributo *pipe*: indica se a alocação alimentará a *tag* Pipe.

A Figura 5.17 apresenta um exemplo de página JSP utilizando a *tag* AllocateWS com os seus parâmetros.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<html>
 <title>Teste da GlueScript</title>
 <body>
  <gs:GlueScript>
    <gs:AllocateWS
      endpoint="http://localhost:8000/DayService/DayService"
      tns="urn:DayService/wsdl"
      portn="DayServiceServantInterfacePort"
      service="DayService"
      method="getDayofWeek"
      parameters="year month day"
      valores="2002 12 20"
      pipe="no"/>
  </gs:GlueScript>
 </body>
</html>
```

FIGURA 5.17 – Utilizando a *Tag* AllocateWS

5.3.5.2 AllocateLOM

A *tag* AllocateLOM é utilizada para invocar um recurso educacional através da localização deste recurso passada como parâmetro.

GlueScriptTags.tld

```
<tag>
  <name>AllocateLOM</name>                                   1
  <tagclass>GlueScript.AllocateLOM</tagclass>
  <bodycontent>JSP</bodycontent>
  <info>
    Verifica se um determinado metadado educacional está disponível.
  </info>
  <attribute>
    <name>name</name>                                        2
    <required>true</required>
  </attribute>
  <attribute>
    <name>lomUrl</name>                                      3
    <required>true</required>
  </attribute>
  <attribute>
    <name>pipe</name>                                        4
    <required>true</required>
  </attribute>
</tag>
```

FIGURA 5.18 – GlueScriptTags - Descrição da *tag* AllocateLOM

A definição da *tag* é apresentada na Figura 5.18. A descrição dos atributos da *tag* são:

1. A definição do nome da *tag* e a classe Java correspondente no pacote GlueScript.

2. Atributo *name:* especifica o nome de um recurso educacional a ser acessado. O campo *required* define se é obrigatório ou não.

3. Atributo *lomUrl:* especifica o endereço de um recurso educacional.

4. Atributo *pipe*: indica se a alocação alimentará a *tag* Pipe

Um exemplo da utilização desta *tag* é apresentado na Figura 5.19.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<%@ page language="java" import="java.util.*" %>
<html>
 <head>
   <title>GlueScript -  Teste LOM (AllocateLOM)</title>
 </head>

 <body>
   <gs:GlueScript>
     <gs:AllocateLOM
        name="The Java Web Services Tutorial"
        lomUrl="http://localhost:8080/jwsdp/docs/tutorial/index.html"
        pipe="no"/>
   </gs:GlueScript>
 </body>
</html>
```

FIGURA 5.19 – Utilizando a *Tag* AllocateLOM

Foi desenvolvida também a *tag* Pipe, cujo objetivo é permitir que um serviço ou recurso possa utilizar como entrada a saída de outro. Esta *tag* é apresentada na seção que segue.

5.3.6 *Tag* Pipe

Esta *tag* tem como objetivo permitir que o resultado da invocação de um *Web Service* ou um recurso educacional possa servir de entrada para outro. A Figura 5.20 mostra a descrição da *tag* Pipe no arquivo descritor de *tags*. São descritos o nome da *tag* e a classe Java correspondente no pacote GlueScript.

```
<tag>
  <name>Pipe</name>
  <tagclass>GlueScript.Pipe</tagclass>
  <bodycontent>JSP</bodycontent>
  <info>
    Conduz um resultado de uma invocação de serviço para outro serviço.
  </info>
 </tag>
```

FIGURA 5.20 – GlueScriptTags - Descrição da *tag* Pipe

A Figura 5.21 apresenta um exemplo de uso da *tag* Pipe. Neste exemplo, após a alocação de um *Web Service*, o resultado é passado a *tag* Pipe para que outro *Web Service* possa utilizar esta saída. Assim como W*eb Services*, os recursos educacionais podem fazer uso da *tag* Pipe. Logo, os dados podem ser manipulados e modificados, adequando-os ao serviço que irá recebê-los como parâmetro de entrada. Caso o desenvolvedor necessite modificar ou converter os tipos de dados de saída para servirem como entrada por outro serviço ou recurso, é possível fazer uso dos recursos disponíveis pelo JSP como, por exemplo, *scriptlets*. O desenvolvedor pode ainda implementar filtros para manipulação destes dados, caso seja de seu interesse, e utilizá-los juntamente com a *tag* Pipe.

Para facilitar o trabalho de localização e alocação de *Web Services* e recursos educacionais foi criado um assistente que será apresentado na próxima seção.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<%@ page language="java" import="java.util.*" %>
<html>
 <title>Teste da GlueScript (Alocação de WS e uso de Pipe)</title>
 <body>
  <gs:GlueScript>
   <gs:Pipe>
    <gs:AllocateWS
     endpoint="http://localhost:8000/MonthService/MonthService"
     tns="urn:MonthService/wsdl"
     portn="MonthServiceServantInterfacePort"
     service="MonthService"
     method="getMonthNumber"
     parameters="language month"
     valores="en december"
     pipe="yes"/>
   </gs:Pipe>
```

```
<%
 String m = (String) pageContext.getAttribute("allocWSpiped");
 String valores = "2002 " + m + " 20";
%>
    <gs:AllocateWS
     endpoint="http://localhost:8000/DayService/DayService"
     tns="urn:DayService/wsdl"
     portn="DayServiceServantInterfacePort"
     service="DayService"
     method="getDayofWeek"
     parameters="year month day"
     valores="<%= valores %>"
     pipe="no"/>
  </gs:GlueScript>
<%
```

FIGURA 5.21 – Utilizando a *Tag* Pipe

5.3.7 Assistente da GlueScript

Como visto anteriormente, as *tags* LocateWS e LocateLOM retornam todos os *Web Services* e recursos educacionais disponíveis em um registro. Desta forma, o trabalho de busca pode tornar-se mais complexo quando um registro apresenta um tamanho muito grande. O retorno de informações referentes aos documentos WSDL pode tornar-se de difícil manuseio. Foi desenvolvido um assistente visando facilitar o trabalho de busca de *Web Services* e recursos educacionais.

Este assistente faz uso das *tags* apresentadas nas seções anteriores. Seu uso é opcional mas permite que o desenvolvedor possa através de registros UDDI e LOM-XML acessar *Web Services* e/ou recursos educacionais de forma simplificada, obtendo uma página JSP com as *tags* básicas.

A Figura 5.22 apresenta o primeiro passo do assistente, o qual permite que o desenvolvedor possa escolher entre localizar e acessar um *Web Service,* ou um recurso educacional.

FIGURA 5.22 – Índice do Assistente

Caso o desenvolvedor opte pela opção de localizar um *Web Service*, a Figura 5.23 apresenta o primeiro passo do assistente, onde o desenvolvedor necessita digitar o endereço de um registro UDDI para que o assistente possa encontrar os documentos WSDLs existentes no registro especificado. O assistente permite que os endereços dos registros UDDIs sejam adicionados ao assistente para uso futuro.

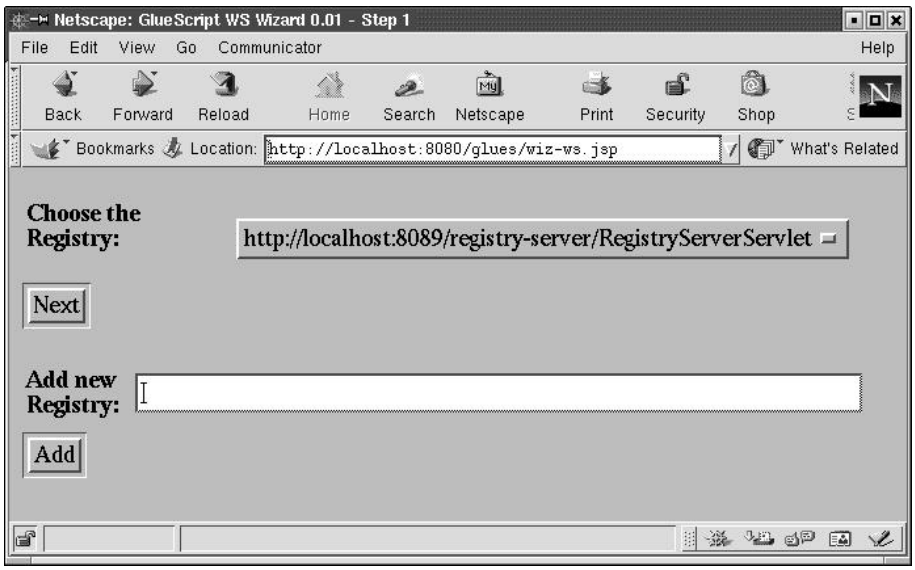

FIGURA 5.23 – Registro UDDI a Ser Acessado

Após o desenvolvedor definir o endereço do registro UDDI e ter clicado o botão *Next*, o assistente retornará ao desenvolvedor uma página (passo 2) com os *Web Services* disponíveis no registro UDDI, conforme a Figura 5.24. A *tag* LocateWS

é utilizada pelo assistente, bem como a *tag* ParsingWS para que os *Web Services* disponíveis possam ser visualizados. O desenvolvedor poderá escolher qual *Web Service* disponível é de seu interesse.

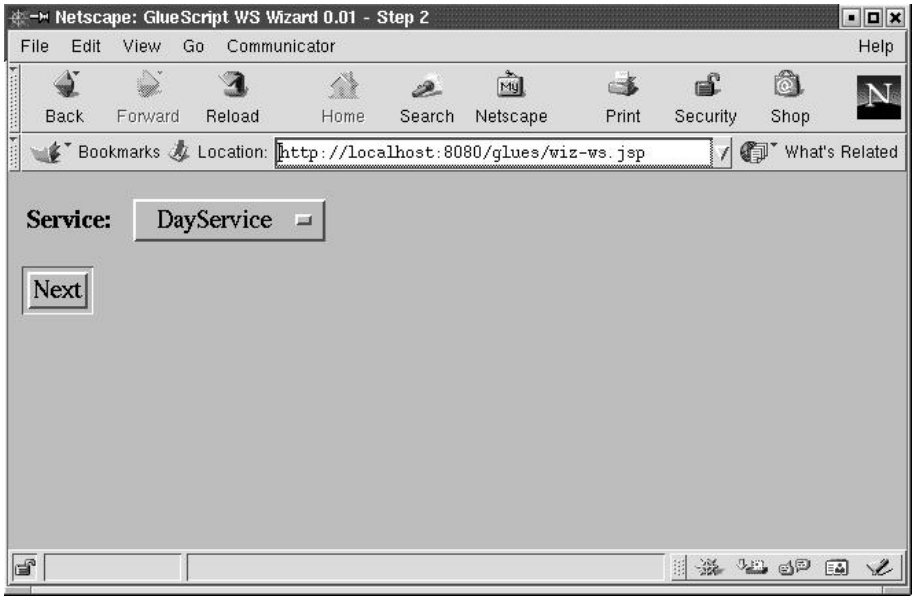

FIGURA 5.24 – Serviços descritos nos documentos WSDL

O assistente disponibiliza para o desenvolvedor, conforme a Figura 5.25, os métodos do *Web Service* que podem ser utilizados. O desenvolvedor escolhe o método e então o assistente no passo 4, retorna os parâmetros do método que devem ser especificados.

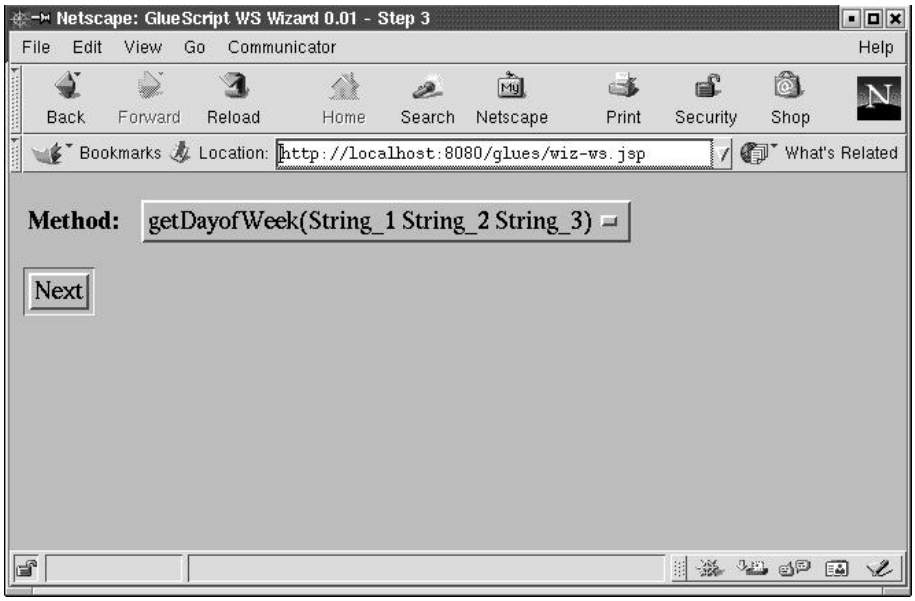

FIGURA 5.25 – Os Métodos do *Web Service*

Os campos dos parâmetros são preenchidos pelo desenvolvedor. A Figura 5.26

apresenta um exemplo de campos de um método escolhido. Os campos representam uma data (ano, mês e dia), que serão utilizados por um *Web Service* que indica qual o dia da semana refere-se a mesma. Os dados dos campos são utilizados pelo assistente para gerar uma página JSP básica com a *tag* LocateWS e os parâmetros necessários que serão enviados para o *Web Service*.

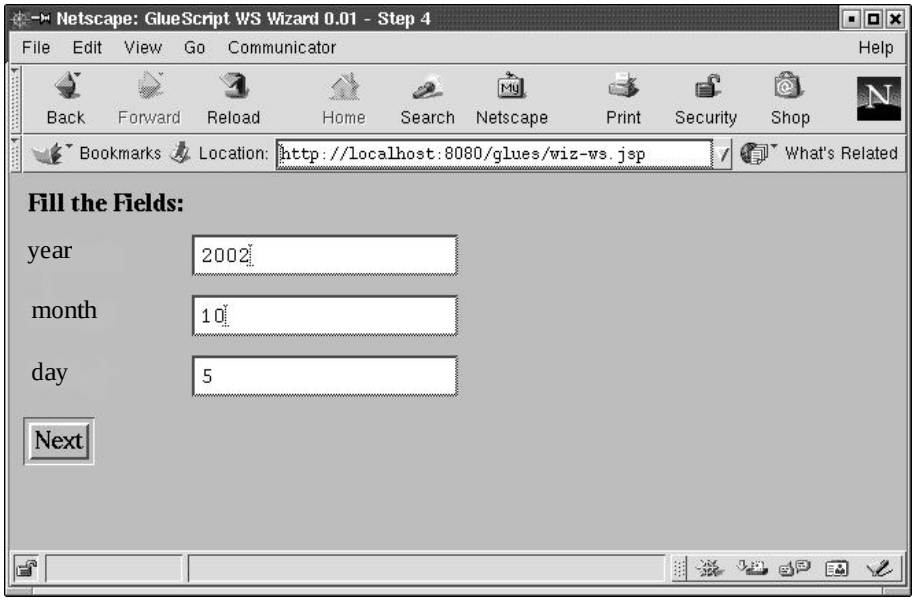

FIGURA 5.26 – Os Parâmetros do Método a Preencher

O arquivo JSP gerado, conforme a Figura 5.27 pode ser visualizado e armazenado através do *link* "*See the GlueScript Source Code*". O código JSP pode ser visualizado no *browser* através do *link* "*Execute the GlueScript Code*". Outra opção é o desenvolvedor é copiar o código fonte visível no campo HTML (TextArea) e compor sua página JSP. Como o assistente aloca apenas um *Web Service* por vez é possível retornar ao início do assistente e recomeçar o processo. Assim será gerado um documento JSP para cada *Web Service* alocado. Desta forma, cabe ao desenvolvedor adequar os códigos fontes em um mesmo arquivo JSP conforme as suas necessidades.

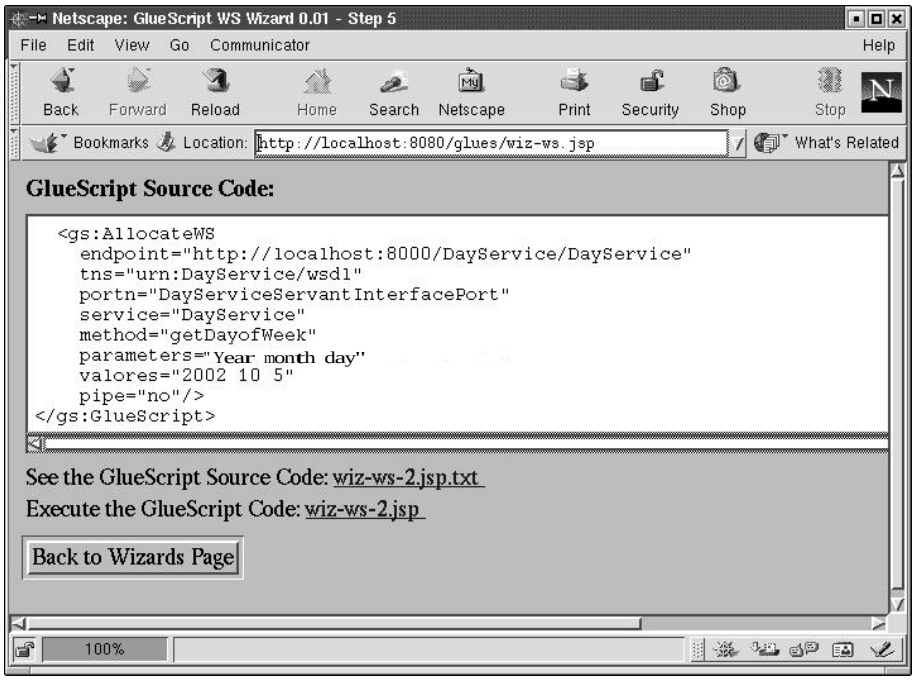

FIGURA 5.27 – Fonte JSP Gerado Pelo Assistente

Caso a escolha do desenvolvedor tenha sido por localizar e acessar um recurso educacional, o assistente apresenta uma página, conforme a Figura 5.28. Neste passo do assistente é necessário especificar a localização do registro de metadados educacionais. Neste passo do assistente é possível adicionar um endereço de um repositório de metadados para utilizá-lo futuramente.

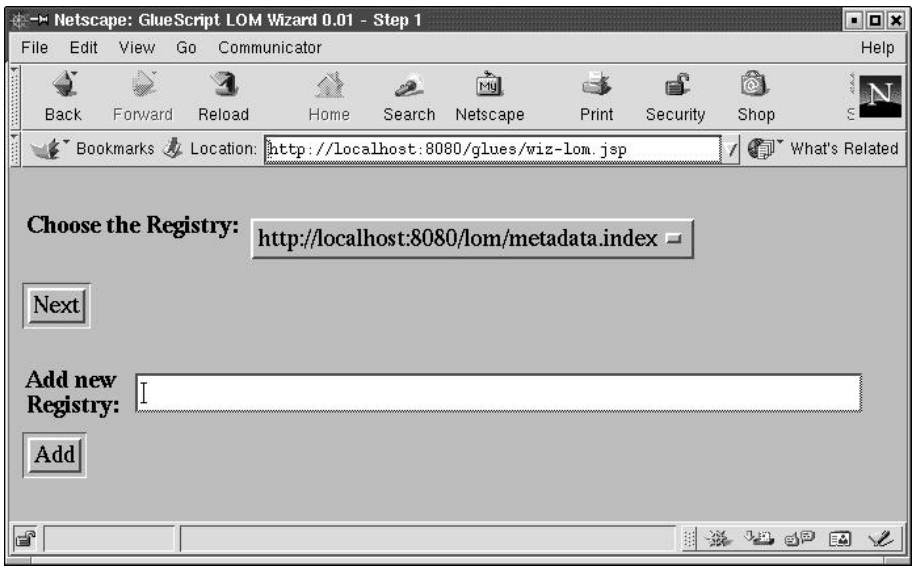

FIGURA 5.28 – Endereço do Registro de Metadados LOM

Após o desenvolvedor especificar o endereço de um registro de metadados LOM-XML e pressionar o botão *Next*, o assistente através das *tags* LocateLOM e

ParsingLOM localiza os metadados e retorna ao desenvolvedor uma página com as opções de metadados disponíveis no registro, conforme a Figura 5.29.

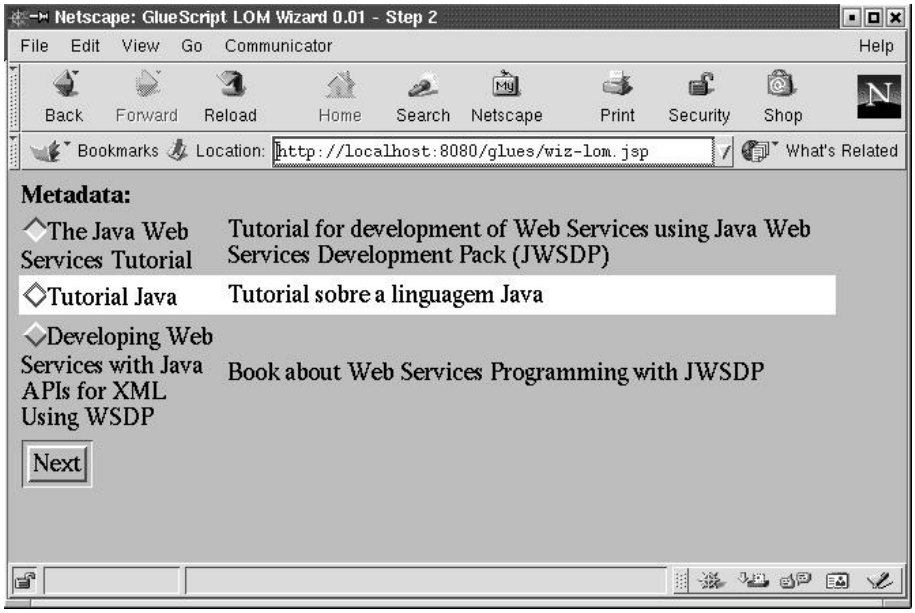

FIGURA 5.29 – Relação dos Recursos Educacionais

O desenvolvedor pode selecionar o recurso educacional e após pressionar o botão *Next*, o assistente gera uma página básica JSP com o alocação do recurso selecionado.

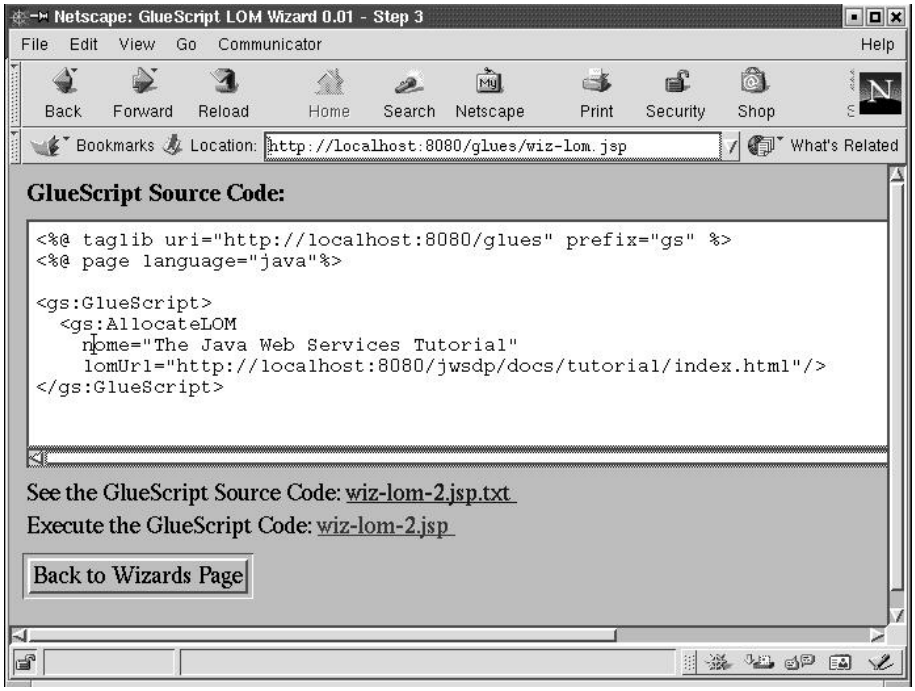

FIGURA 5.30 – Código Fonte JSP do Recurso Educacional Alocado

A Figura 5.30 apresenta um exemplo básico de página JSP gerada com a *tag* AllocateLOM e seus parâmetros, para que o recurso possa ser então alocado. Da mesma forma como o mencionado no assistente para *Web Services*, o desenvolvedor pode reiniciar o processo do assistente. Como para cada recurso educacional alocado é criado um novo documento JSP, é de responsabilidade do desenvolvedor adequar o código as suas necessidades.

5.4 Comparando a GlueScript

Conforme foi apresentado nas seções anteriores, a linguagem GlueScript apresenta características muito particulares. Para que estas características se tornem mais claras, a Tabela 5.1 apresenta um comparativo entre a GlueScript e as demais linguagens de composição mencionadas no início deste capítulo.

As principais características da GlueScript estão em utilizar o conceito de *Web Services* através do uso de WSDL, SOAP e UDDI. Também por endereçar objetos LOM além de permitir acesso de forma dinâmica. O fato da linguagem não permitir coersão de dados pode dificultar composição quando dados necessitarem ser modificados. Porém por ser extensível, esta é uma característica que pode ser acrescentada à GlueScript.

Características	Piccola	WebL	Mawl	WebCompose	GlueScript
Utiliza Conceito de *Web Services*	Não	Não	Não	Não	Sim
Utiliza WSDL, SOAP e UDDI	Não	Não	Não	Não	Sim
Endereça Objetos LOM	Não	Não	Não	Não	Sim
É Extensível	Sim	Sim	Sim	Sim	Sim
Embeded HTML	Sim	Sim	Sim	Sim	Sim
Permite Coersão	Não	Não	Não	Sim	Não
Formato de Execução	*Server Side*	CGI	CGI	CGI	*Server Side*
Acesso Dinâmico	Não	Não	Não	Não	Sim

TABELA 5.1 – Comparativo Entre as Linguagens de Composição

5.5 Conclusão

Através do uso das *tags* da GlueScript torna-se possível a localização e acesso de forma dinâmica de *Web Services* e recursos educacionais. As *tags* desenvolvidas permitem a composição destes *Web Services* e recursos educacionais, de forma que estes possam gerar novas aplicações Web.

Foi criado um assistente que utiliza as *tags* desenvolvidas. O assistente tende a facilitar a localização e acesso de *Web Services* e recursos educacionais tornando este trabalho mais simplificado.

Como visto neste capítulo é possível gerar aplicações Web utilizando apenas *Web Services,* ou recursos educacionais, ou ambos. Com o objetivo de testar a GlueScript, o próximo capítulo apresenta um estudo de caso, o qual utiliza as t*ags* desenvolvidas.

Capítulo 6

Estudo de Caso

Resumo

Este capítulo apresenta um estudo de caso de uso das *tags* da GlueScript com a finalidade de verificar a usabilidade da linguagem, além de verificar eventuais restrições no uso da mesma.

6.1 Introdução

Com o objetivo de avaliar as funcionlidades das *tags* da GlueScript, foi realizado um estudo de caso utilizando as *tags* desenvolvidas para permitir composição de *Web Services* e recursos educacionais.

O estudo de caso se baseia na construção de uma página de um curso baseado na Web. Este curso apresenta momentos síncronos e assíncronos e o objetivo é verificar as vantagens e desvantagens no uso da GlueScript. Desta forma as próximas seções apresentam como seria o trabalho de um desenvolvedor em criar um curso sem o uso das *tags* da GlueScript e como o uso da linguagem poderia facilitar a construção da página.

6.2 Construindo um Curso na Web

Quando alguém deseja construir um curso na Web com momentos síncronos e assíncronos torna-se necessário alguns conhecimentos, tais como:

- Conhecer uma linguagem de programação que tenha suporte para Web. Esta linguagem deve permitir a construção de aplicações como *chat*, fórum, lista de discussão dentre outras.
- A partir do conhecimento de uma linguagem, um desenvolvedor precisa projetar como será a página do seu curso.
- Após ter definido a estrutura que seu curso terá, torna-se necessário a implementação de todas as aplicações a serem utilizadas.

Na construção de um curso para Web utilizando as vias comuns, um desenvolvedor muito remotamente tem a possibilidade de reutilização de código de aplicações. Além disso, há uma demanda de tempo muito grande, já que é necessário implementar todas as aplicações.

6.3 Construindo um Curso na Web com a GlueScript

Um desenvolvedor ao utilizar a GlueScript na criação de um curso para Web, obtém benefícios tais como:

- Não é necessário ter conhecimento de nenhuma linguagem de programação, já que não há necessidade de implementar todas as aplicações que farão parte da página do curso.
- O desenvolvedor após definir a estrutura do curso, utiliza a GlueScript para localizar os *Web Services* e recursos educacionais de que necessita.
- Estes *Web Services* e recursos educacionais são acessados de forma a compor uma página de curso com aplicações já existentes que estão disponíveis na Web.
- Para que o desenvolvedor possa criar interfaces aos usuários do ambiente Web, torna-se necessário o conhecimento de HTML e JSP.

Assim, os pré-requisitos necessários para utilização da GlueScrit são:

- Conhecimento de Tomcat;
- HTML e JSP são necessários para desenvolver as interfaces.

6.4 Estudo de Caso

Como estudo de caso, foi desenvolvida uma página de um curso de *Web Services*. A página de curso foi criada por um profissional da área de redes, com experiência em linguagens de programação e desenvolvimento de aplicações diversas. O desafio deste profissional estava em desenvolver uma página de curso com uso das *tags* da GlueScript. O curso foi estruturado a partir de *Web Services* e recursos educacionais disponíveis na Web. A página do curso apresenta uma dinâmica de trabalho com momentos síncronos (*chat*) e assíncronos (acesso a página, *e-mail*).

As páginas de *Chat* e *EMail* foram construídas através da invocação de *Web Services* já existentes. O tutorial e o livro disponibilizados na página de Apostilas e Livros foram integrados ao curso através da especificação de LOM-XML.

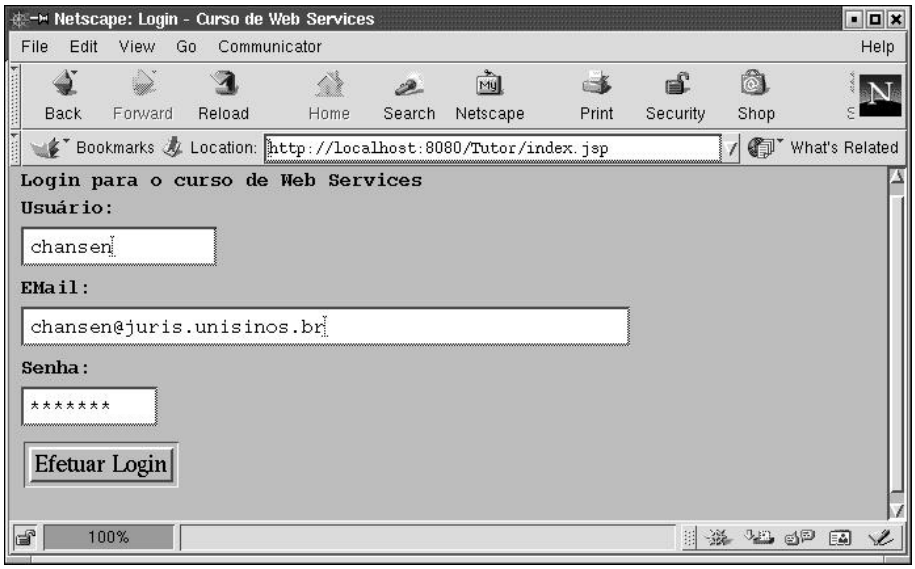

FIGURA 6.1 – Página de Autenticação do Usuário

É preciso que um usuário efetue *login* para que possa ter acesso à página do curso. A Figura 6.1 apresenta a página de *login* na qual é necessário que o usuário entre com seu nome, endereço de *e-mail* e senha.

A Figura 6.2 apresenta a página principal do curso.

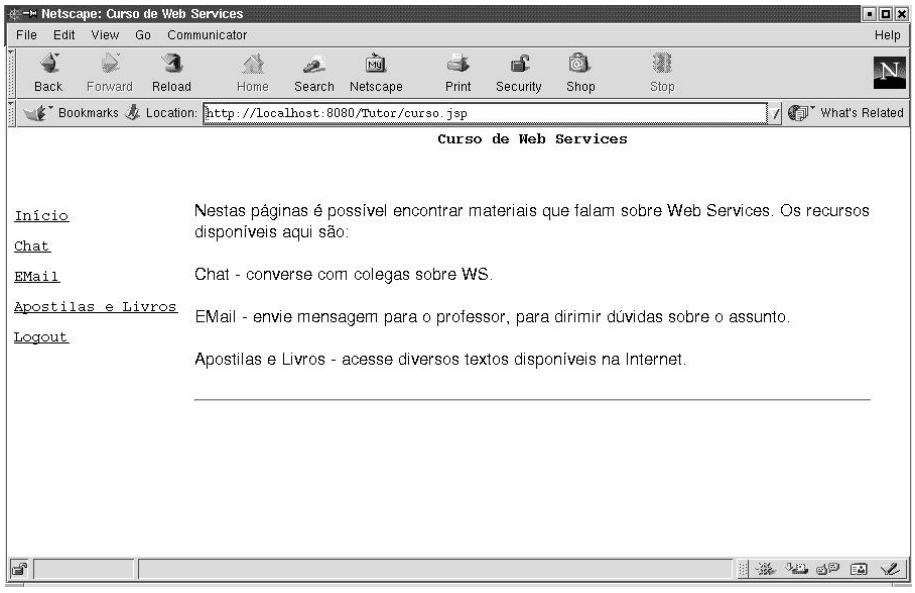

FIGURA 6.2 – Página Principal do Curso

A partir da página principal o aluno pode escolher entre enviar um *e-mail* ao professor, acessar o *chat* e trocar idéias a fim superar possíveis dúvidas, além de ter acesso a tutoriais e livros.

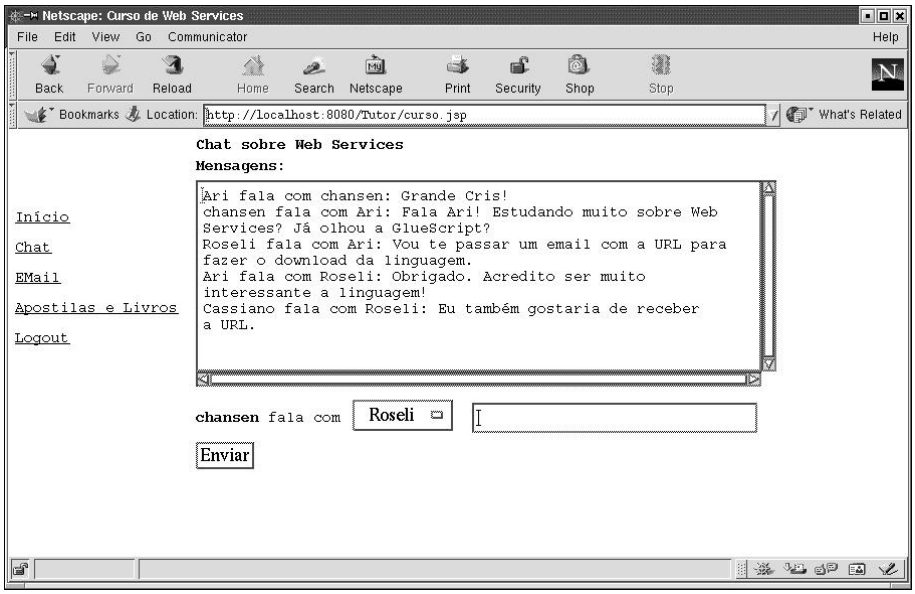

FIGURA 6.3 – Página de *Chat* do Curso

Através do link *Chat*, o aluno acessa uma página conforme a Figura 6.3, na qual pode conversar sobre *Web Services* com demais interessados no assunto, que tenham acesso ao curso.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<%@ page language="java" import="java.util.*" %>
<html>
 <title>Chat</title>
 <body>

  <gs:GlueScript>
   <gs:AllocateWS
    endpoint="http://localhost:8000/ChatService/ChatService"
    tns="urn:ChatService/wsdl"
    portn="ChatServiceServantInterfacePort"
    service="ChatService"
    method="sendMessage"
    parameters="from to message"
<%
 String valores = request.getParameter("from") + " " +
                  request.getParameter("to") + " " +
                  request.getParameter("msg");

%>
    valores="<%= valores %>"
    pipe="no"/>
  </gs:GlueScript>

 </body>
</html>
```

FIGURA 6.4 – Código da Página do *Chat*

A Figura 6.4 apresenta o código JSP com a *tag* AllocateWS a qual faz a alocação do *Web Service ChatService*.

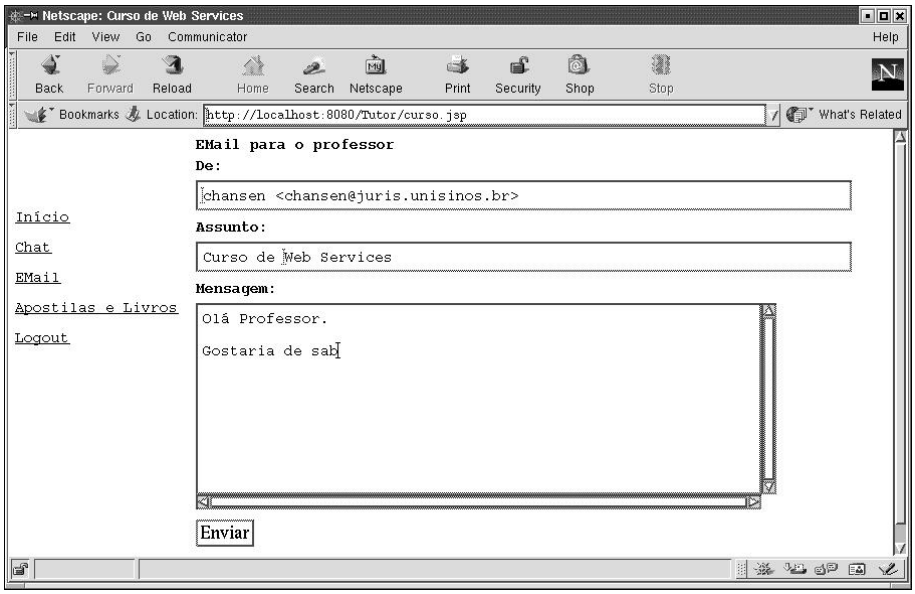

FIGURA 6.5 – Página de E-mail do Curso

Ao acessar o link *EMail* o aluno pode enviar *e-mails* ao professor do curso, de forma a obter infomações adicionais ou sanar dúvidas que possa ter. A Figura 6.5 apresenta a página que permite o envio de mensagens.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<%@ page language="java" import="java.util.*" %>
<html>
 <title>EMail</title>
 <body>

   <gs:GlueScript>
     <gs:AllocateWS
       endpoint="http://localhost:8000/EMailService/EMailService"
       tns="urn:EMailService/wsdl"
       portn="EMailServiceServantInterfacePort"
       service="EMailService"
       method="sendEMail"
       parameters="from to subject SMTPserver message"
<%
 String valores = request.getParameter("from") + " " +
                  request.getParameter("to") + " " +
                  request.getParameter("subject") + " " +
                  request.getParameter("smtp") + " " +
                  request.getParameter("msg");

%>
       valores="<%= valores %>"
       pipe="no"/>
   </gs:GlueScript>

 </body>
</html>
```

FIGURA 6.6 – Código da Página de E-mail

O código JSP correspondente é apresentado na Figura 6.6. A *tag* AllocateWS é utilizada para invocação do serviço de *e-mail*.

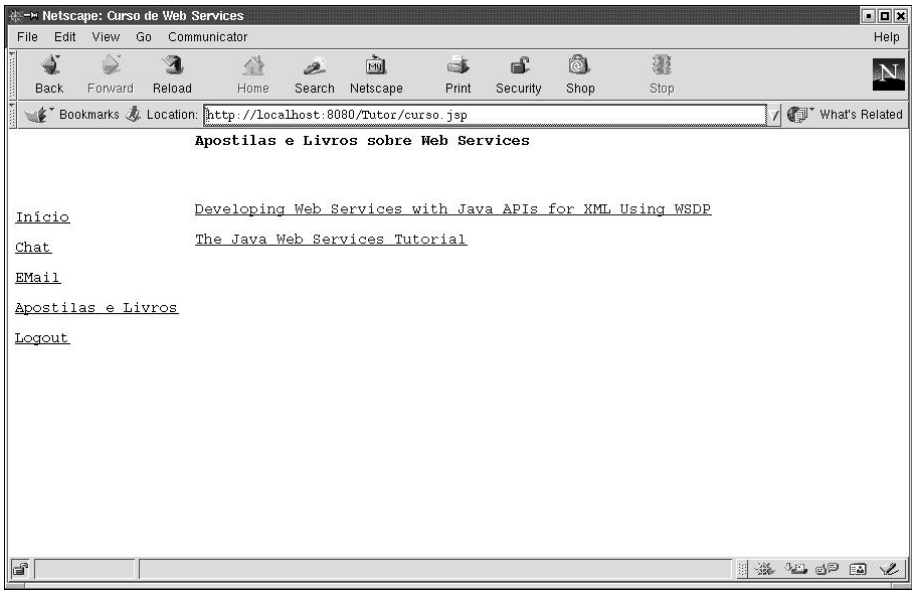

FIGURA 6.7 – Página de Apostilas e Livros do Curso

Caso seja do interesse do aluno, o curso disponibiliza uma página com *links* a apostilas e tutoriais sobre *Web Services*. Na Figura 6.7 são apresentados dois *links* de um livro e um tutorial sobre *Web Services*.

O *link* do livro *Developing Web Services With Java APIs for XML Using WSDP* apresenta o código JSP com uso da *tag* AllocateLOM conforme a Figura 6.8.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<%@ page language="java"%>

<gs:GlueScript>
 <gs:AllocateLOM
   name="Developing Web Services with Java APIs for XML Using WSDP"
   lomUrl="http://kelsen.unisinos.br/~chansen/ebook.pdf"/>
</gs:GlueScript>
```

FIGURA 6.8 – Código do *Link* do Livro

O *link* do tutorial *The Java Web Services Tutorial*, apresenta o código JSP conforme a Figura 6.9 que também faz uso da *tag* AllocateLOM.

```
<%@ taglib uri="http://localhost:8080/glues" prefix="gs" %>
<%@ page language="java"%>

<gs:GlueScript>
 <gs:AllocateLOM
   name="The Java Web Services Tutorial"
   lomUrl="http://localhost:8080/jwsdp/docs/tutorial/index.html"/>
</gs:GlueScript>
```

FIGURA 6.9 – Código do *Link* do Tutorial

A partir da construção deste curso, foi possível entrevistar o desenvolvedor e obter informações relevantes sobre o trabalho realizado. Estas informações são relacionadas como segue:

- O desenvolvedor tinha pouca experiência com o desenvolvimento de cursos Web.
- Tinha pouco conhecimento em *Web Services*.
- Não tinha conhecimento em LOM-XML.
- Conhecia JSP e servidor Tomcat.
- Considerou o assistente uma ferramenta muito útil para o desenvolvimento da página do curso, tendo facilitado a localização e acesso aos serviços e recursos desejados.

Através das informações fornecidas pelo desenvolvedor, foi possível observar que as *tags* da GlueScript desenvolvidas tendem a facilitar a construção de páginas Web, através da invocação de Web Services e recursos educacionais. Porém, algumas restrições foram observadas os quais serão descritas a seguir.

6.5 Restrições ao Uso da GlueScript

Algumas restrições foram detectadas através do uso da GlueScript. Inicialmente observou-se que alguns registros públicos de *Web Services* disponíveis na Web, como por exemplo o da IBM, apresentavam um número muito grande de *Web Services* com direcionamento de uso a máquinas locais. Além disso, muitos dos *Web Services* disponíveis não apresentavam funcionalidades relevantes. Alguns *Web Services* são apenas testes de alguns desenvolvedores em verificar apenas a funcionalidade de um registro UDDI e não de disponibilizar *Web Services* para uso no registro.

Outra forma de uso da GlueScript é através de registros UDDIs locais. Foi observado um *bug* no conjunto de APIs da Sun (JWSDP) o qual foi usado no desenvolvimento das *tags*, que não permitia a publicação de um *Web Service* em um registro local de maneira funcional. Desta forma, para fim de testes, foi necessária a instalação do produto comercial Sun ONE Studio da Sun Microsystems (versão 60 dias) e assim fosse possível trabalhar com registros UDDIs locais.

Foi observado também que os registros de recursos educacionais até o momento não são padronizados. Apesar da descrição dos recursos seguir um padrão e facilitar o acesso a estes recursos, a não padronização dos repositórios de metadados dificulta o acesso a estes recursos dinamicamente. No Capítulo 3 foi mencionada a preocupação do projeto IMS em padronizar os repositórios, o que permitirá maior integração entre os recursos educacionais.

Outra restrição das *tags* da GlueScript está em não ter suporte a tipos complexos de dados. Alguns *Web Services* utilizam, por exemplo, vetores ou outras estruturas mais complexas. A versão atual da GlueScript não abrange o uso desses tipos de dados.

6.6 Conclusão

Este capítulo apresentou um estudo de caso de uso das *tags* da GlueScript. Foi possível observar que as *tags* tendem a facilitar o trabalho de criação de páginas Web. O fator relevante na utilização das *tags* está em permitir a composição de *Web Services* e recursos educacionais de forma que uma página Web pode ser construída a partir de serviços e recursos já existentes.

Foram encontradas algumas restrições ao uso da GlueScript, porém sendo esta uma linguagem extensível, estas restrições podem ser solucionadas. Outro fator relevante no desenvolvimento do estudo de caso, foi a utilização do assistente da GlueScript para realizar a localização dos *Web Services* e recursos educacionais. O assistente demonstrou ser uma ferramenta útil, tornando o trabalho do desenvolvedor mais ágil.

Capítulo 7

Conclusões e Trabalhos Futuros

7.1 Conclusões

Esta dissertação apresentou o problema de criação de ambientes para a Web. Esta é uma tarefa que dispende conhecimentos específicos de linguagens com suporte a aplicações Web, além de tempo na construção de aplicações necessárias. Como forma de amenizar este problema, este trabalho sugere uma arquitetura e uma linguagem de composição baseada em *Web Services* para a criação destes ambientes.

Algumas tecnologias foram apresentadas nesta dissertação. Por exemplo, uma visão geral de *Web Services* e um estudo sobre metadados educacionais.

A utilização da tecnologia baseada em *Web Services* permite que serviços possam ser disponibilizados na Web de forma padronizada. Esta padronização permite a localização e acesso a serviços de forma bastante eficiente, já que as tecnologias empregadas estão baseadas em XML. O uso do protocolo SOAP facilita o acesso a estes serviços de forma que o mesmo é utilizado junto com HTTP. As mensagens SOAP são transmitidas em formato XML, o que facilita a manipulação de dados de entrada e saída.

Os metadados permitem a padronização da descrição de recursos educacionais. Foram estudados metadados como o *Dublin Core* e o LOM. Optou-se a utilizar o metadado LOM-XML por ser um dos mais utilizados e resultado de esforços de projetos como o IMS e ARIADNE.

Cabe salientar, porém, a dificuldade em utilizar a GlueScript devido a falta de *Web Services* funcionais em registros UDDI. Desta forma, foi necessária a construção de diversos *Web Services* para a realizaçao dos testes, o que conseqüentemente dispendeu um trabalho maior do que o previsto.

Por não haver até o momento nenhum trabalho similar na área, não foi possível comparar o trabalho desenvolvido com trabalhos correlatos.

7.2 Contribuições

- Foi possível propor uma arquitetura de integração das tecnologias de *Web Services* e recursos educacionais descritos através de LOM-XML. Esta integração se dá através da linguagem de composição GlueScript.

- A linguagem GlueScript é uma *Tag Library* JSP, dando suporte à localização e acesso a *Web Services* e recursos educacionais. Através do uso das *tags*

da GlueScript a construção de ambientes Web torna-se facilitada pois um desenvolvedor não necessita implementar as aplicações de que necessita.

- Ao utilizar a GlueScript um desenvolvedor pode também fazer uso da linguagem JSP para dar mais poder de processamento a linguagem. O uso de JSP em conjunto com a GlueScript permite que o desenvolvedor possa acrescentar funcionalidades à sua página. As *tags* da GlueScript têm funcionalidades específicas para localização e acesso a *Web Services* e recursos educacionais. Caso o desenvolvedor deseje trabalhar com banco de dados, por exemplo, este deverá recorrer às *tags* específicas de JSP. Outra possibilidade na utilização de JSP associada à GlueScript é permitir ao desenvolvedor que conhece a linguagem de programação Java, adicionar *scriptlets* dentro da página JSP.

- Através do uso das *tags* da GlueScript é possível localizar e acessar *Web Services* e recursos educacionais de forma dinâmica.

- O estudo de caso foi realizado utilizando-se as *tags* da GlueScript no qual foi possível observar as facilidades na localização e acesso a *Web Services* e recursos educacionais.

7.3 Trabalhos Futuros

A seguir serão listados alguns trabalhos que poderão ser realizados como continuidade desta dissertação.

- O Capítulo 3 desta dissertação menciona a inciativa para padronização dos repositórios de metadados educacionais. A iniciativa do projeto IMS tornará os repositórios interoperáveis. Para dar suporte a esta padronização a GlueScript necessitará ser adaptada.

- A linguagem GlueScript não tem suporte à manipulação de tipos complexos. Para que isso seja possível é necessário aperfeiçoar o conjunto de *tags*.

- A *tag* Pipe necessita ser aperfeiçoada de forma a permitir a transformação de dados (filtros) utilizados como entrada e saída para *Web Services* e recursos educacionais. Até o momento esta *tag* apenas direciona as saídas e entradas de dados para os serviços e recursos.

Bibliografia

[1] Achermann, F.; Lumpe, M.; Schneider, J. and Nierstrasz, O. *PICCOLA a Small Composition Language.* In Formal Methods for Distributed Processing - A survey of Object-Oriented Aproaches, Howard Bowman and John Derrick(Eds.), pp. 403-426, Cambridge University Press, 2001.

[2] Amrstrong, E.; Bodoff, S.; Carson, D.; Fisher, M.; Green, D. and Haase, K. *The Java Web Services Tutorial.* http://java.sun.com/webservices/dowloads/webservicestutorial.html. Último acesso - Abril 2002.

[3] Anido, L. E.; Fernández, M. J.; Caeiro, M.; Santos J. M.; Rodríguez, J. S.; Llamas, M. *Educational metadata and brokerage for learning resources.* Computer & Education. Publish by Elsevier Science Ltd. December 2001. In Press.

[4] ARIADNE *Educational Metadata Recommendation.* December 1999. http://ariadne.unil.ch/. Último acesso - Maio 2002.

[5] ARIADNE *Educational Metadata Recommendation*, Version 3.2. February 2002. http://ariadne.unil.ch/. Último acesso - Maio 2002.

[6] ARIBA, INC., International Business Machines Corporation and Microsoft Corporation. *UDDI Data Structures Reference.* September, 2000.

[7] Artesia. http://www.artesia.com. Último acesso - Maio 2002.

[8] Artesia-TEAMS. http://www.artesia.com/teams_arquitecture.html. Último acesso - Maio 2002.

[9] Atkins, D. L.; Ball, T.; Bruns, G. and Cox, K. *Mawl: A Domain-Specific Language for Form-Based Services.* IEEE Transactions on Software Engineering, Vol. 25 Number 3, May/Jun 1999.

[10] Babu, S. C. *e-Learning Standards.* 2001.

[11] BECTa - British Educational Communications and Technology agency. *Metadata in Education.* August 2001.

[12] Box, D; Ehneubuske, D.; Kakivaya, G.; Layman, A.; Mendelsohn, N; Nielsen, h. F.; Thatte, S.; Winer, D. *SOAP: Simple Object Access Protocol*, April 2000.

[13] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M. and Maler, E. eXtensível Markup Language (XML) 1.0. Second Edition. http://www.w3.org/TR/2000/REC-xml-20001006. Último acesso - October 2000.

[14] Brittenham, P.; Cubera, F.; Ehnebuske, D.; Graham, S. *Understanding WSDL in a UDDI registry, Part 1*, September 2001.

[15] Brittenham, P. *Understanding WSDL in a UDDI registry, Part 2*, September 2001.

[16] Cardelli, L. and Davies, R. *Service Combinators for Web Computing*. IEEE Transaction on Software Engineering, vol. 25 number 3, May/Jun 1999.

[17] Cheng, J. and Xu, J. *IBM DB2 XML extender*. February 2000 www-4.ibm.com/software/data/db2/extenders/xmlext/xmlextbroch.pdf. Último acesso - Abril 2002

[18] Clarke, N. W. *Silverstream and Web Services -A Technical Overview & Developer's Perspective of SOAP*, June 2000.

[19] Clements, T. *Overview of SOAP Web Services - Technical Overviews*. Sun Microsystems, August 2001.

[20] Christensen, E.; Curbera, F.; Meredith, G. and Weerawarana, S. *Web Services Description Language (WSDL)*, March 2001.

[21] *Dublin Core Metadata Element Set, Version 1.1: Reference Description*. July 1999. http://dublincore.org/documents/1999/07/02/dces/. Último acesso - Maio 2002.

[22] Durm, R. V.; Duval, E.; Verhoeven, B.; Cardinaels, K.; Olivié, H. *ARIADNE: A Modular Open Learning Platform*. October 2000.

[23] EdNA *Education Network Australia*. http://www.edna.edu.au. Último acesso - Maio 2002.

[24] Ehnebuske, D.; Rogers, D.; Riegen, C. V. *UDDI Version 2.0 Data Structure Reference*. June 2001.

[25] Fields, D. K.; Kobb, M. A. *Web Development With Java Server Pages*. Manning Publications Co. 2000. 554 pages.

[26] Forte, E.; Haenni F.; Warkentyne, K.; Duval, E.; Cardinaels, K.; Vervaet, E.; Hendrikx, K.; Forte, M. W.; Simillion, F. *Semantic and Pedagogic Interoperability Mechanisms in the ARIADNE Educational Repository*. SIGMOD, March 1999, Vol 28, No 1.

[27] Forte, E.; Haenni F.; Warkentyne, K.; Duval, E.; Cardinaels, K.; Hendrikx, K.; Forte, M. W.; Verhoeven, B.; Durm, R. V.; Ebel, N.; Macowicz, M. *The ARIADNE Knowledge Pool System*. Communications of the ACM, May 2001, Vol. 44, No. 5.

[28] Foster, P.; Kraner, M.; Graziano, A.; Romano, S. P. *GESTALT - Getting Educational Systems Talking Across Leading-edge Technologies.* April 2000.

[29] GEM *Gateway to Educational Materials.* http://www.geminfo.org. Último acesso - Maio 2002.

[30] Graham, S.; Simeonov, S.; Boubez, T.; Davis, D.; Daniels, G.; Yuichi, N. and Neyama, R. *Building Web Services With Java.* SAMS, 2002. 581 pages.

[31] Gudgin, M. ; Handley, M.; Moreau, J. and Nielsen, H. F. *SOAP Version 1.2 Part 1: Messaging.* W3C, October 2001.

[32] Gudgin, M. ; Handley, M.; Moreau, J. and Nielsen, H. F. *SOAP Version 1.2 Part 2: Adjuncts.* W3C, October 2001.

[33] Gudgin, M.; Hadley, M. Moreau, J.; Nielsen, H. F. *SOAP Version 1.2.* W3C, July 2001.

[34] Hansen, R. P.; Santos, C.T.; Pinto, S. C. C. S.; Lanius, G. L and Massen, F. *Web Services: An Architectural Overview.* First International Seminar on Advanced Research in E-Business - EBR 2002. PUC-RIO. November 2002.

[35] Hanson, J. *Use Web services to integrate web applications with EISs.* http://www.javaworld.com/javaworld/jw-02-2002/jw-0208-eis_p.html. Último acesso - Fevereiro 2002.

[36] Hall, M. *Core Servlets and Java Server Pages.* Sun Microsystems. 589 pages.

[37] Hendricks, M.; Galbraith, B.; Irani, R.; Milbery, J.; Modi, T.; Tost, A.; Toussaint, A.; Basha, S. J. and Cable, S. *Professional Java Web Services.* Wrox, January 2002. 588 pages.

[38] Hillmann, D. *Using Dublin Core.* April 2001. http://dublincore.org/documents/2001/04/12/usageguide/. Último acesso - Maio 2002.

[39] *HP Web Services Platform Architecture - an overview.* October, 2001. http://www.bluestone.com/downloads/pdf/web_services_architecture.pdf. Último acesso - Março 2002.

[40] IEEE LTSC. *Draft Standard for Learning Object Metadata.* March 2002. http://ltsc.ieee.org/. Último acesso - Maio 2002.

[41] IMS *Global Learning Consortium*, Inc. *IMS Learning Resource MetaData Information Model.* September 2001. http://www.imsproject.org/. Último acesso - Maio 2002.

[42] IMS *Global Learning Consortium*, Inc. *IMS Learning Resource Meta-Data Best Practice and Implementation Guide.* September 2001. http://www.imsproject.org/. Último acesso - Maio 2002.

[43] IMS *Global Learning Consortiu*m, Inc. *IMS Learning Resource Meta-Data XML Binding.* September 2001. http://www.imsproject.org/. Último acesso - Maio 2002.

[44] IMS *Digital Repositories White Paper.* Version 1.6, August 2001. http://www.imsproject.org/. Último acesso - Maio 2002

[45] IMS *Global Learning Consortium*, INC. http://www.imsproject.org/. Último acesso - Maio 2002.

[46] Kao, J. *Developers' Guide to Building XML-based Web Services with the java 2 Platform, Enterprise Edition (J2EE*). Sun Microsystem, Inc. June 2001.

[47] Kraft, Reiner. *How IBM WebSphere Studio Application Developer Compares with Microsoft Visual Studio .NET - Part1: Conceptual Differences.* http://www7b.boulder.com/wsdl/techjournal/0202_kraft/kraft.html. Último acesso - Fevereiro 2002.

[48] Kreger, H. *Web Services Conceptual Architecture.* IBM Software Group, May 2001.

[49] Kurniawan, B. *Java for the Web with Servlets, JSP and EJB.* New Riders. 2002. 953 pages.

[50] Lucena, C.; Fuks, H. *A Educação na Era da Internet.* Clube do Futuro. 2000. 156 páginas.

[51] Markus, B. *Educational Metadata.* 2000

[52] McLaughlin, B. *Java and XM*L. O'Reilly, 2001, 528 pages.

[53] Microsoft Corporation. *XML WEb Services - A new Opportunity for ISV*s. http://www.microsoft.com/serviceproviders/strategicwp/xml_web_services.asp, 2001. Último acesso - Março 2002.

[54] Morais, H. *Compaq's Web Language. A Programing Language for the Web.* Compaq Systems Research Center (SRC). 1998-1999.

[55] NSDL *National SMETE Digital Library.* http://www.nsdl.nsf.gov/index.html. Último Acesso - Maio 2002.

[56] Pinto, S. C. C. S; Fontoura, M. F. and Lucena, C. J. *A Web-based Educational Environments Comparison using a Conceptual Model compatible with the EDUCOM/IMS Platform.* Brazilian Symposium on Education and Computer Science (SBIE'98), Fortaleza, Brazil, 1998 (in Portuguese).

[57] Pinto, S. C. C. S. *Composição em WebFrameworks.* Tese de doutorado. Pontífica Universidade Católica do Rio de Janeiro. Agosto, 2000.

[58] Rajaram A. *Overview of UDDI. Web Services - Tecnical Overviews.* Sun Microsystems, August 2001.

[59] Robson, R. *Report on Learning Technology Standards.* ED-Media 2000.

[60] Schachor, G.; Chace, A. and Rydin M. *JSP Tag Libraries.* Manning Publications Co. 2001. 623 pages.

[61] Seely, S. *SOAP Cross Platform Web Service Development Using XML.* Prentice Hall PTR, 2002. 391 pages.

[62] Silva, W. L. S. *JSP and Tag Libraries for Web Development.* New Riders. 2002. 442 pages.

[63] *The World's Leading Resource for Internet Trends &Statistics.* www.nua.com. Último acesso - Fevereiro2003.

[64] Vawter, C. and Roman, E. *J2EE vs. Microsoft.NET - A Comparision of Building XML-based Web Services*, June 2001.http://www.midleware-company.com/. Último acesso - Março 2002.

[65] Wahli, U.; Tomlinson, M.; Zimmermann, O.; Deruyck, W. and Hendriks, D. *Web Services Wizards with WebSphere Studio Application Developer. Redbooks* - IBM. April 2002, 598 pages.

[66] Wolter, R. *XML Web Services Basics.* Microsoft Corporation, December 2001.http://msdn.microsoft.com/library/default.asp?url=/library/en-us/Dnwebsrv/html/webservbasics.asp?frame=true. Último acesso - Março 2002.

[67] Yin, K. *A Reference Architecture for Smart Web Services.* Sun Microsystems, August 2001.