MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-14609-TDI/1189

WISS-SERVIÇO WEB PARA SEGMENTAÇÃO DE IMAGENS: ESPECIFICAÇÃO E IMPLEMENTAÇÃO

Luigi Carli Marroni Aulicino

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelo dr. Antônio Miguel Vieira Monteiro, aprovada em 03 de abril de 2006.

INPE
São José dos Campos
2007

681.03.6 : 528.711.7

Aulicino, L. C. M.
WISS – serviço web para segmentação de imagens: especificação e implementação / Luigi Carli Marroni Aulicino. - São José dos Campos: 2006.
116p. ; (INPE-14609-TDI/1189)

1.Serviço web. 2.Terralib. 3.MapServer. 4.OGC. 5.Serviço web de mapas (WMS). 6.Serviço web para segmentação de imagens (WISS). I.Título.

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de **Mestre** em

Computação Aplicada

Dr. Rafael Duarte Coelho dos Santos

Presidente / INPE / SJCampos - SP

Dr. Antonio Miguel Vieira Monteiro

Orientador(a) / INPE / SJCampos - SP

Dra. Leila Maria Garcia Fonseca

Membro da Banca / INPE / SJCampos - SP

Dr. Gilberto Câmara

Membro da Banca / INPE / SJCampos - SP

Dr. Jugurta Lisboa Filho

Convidado(a) / UFV / Viçosa - MG

Aluno (a): **Luigi Carli Marroni Aulicino**

São José dos Campos, 03 de abril de 2006

“*As grandes idéias são aquelas nas quais a única coisa que nos surpreende é que não nos tivessem ocorrido antes*.”

Noel Clarasó, escritor espanhol.

AGRADECIMENTOS

Chegou à hora de agradecer àquelas pessoas que, de alguma forma, contribuíram para a realização deste trabalho. Pessoas que me incentivaram de diferentes maneiras. A todos vocês o meu sincero “MUITO OBRIGADO”!

Ao Instituto Nacional de Pesquisas Espaciais (INPE), pela oportunidade de fazer o mestrado.

Aos professores da Computação Aplicada (CAP) e da Divisão de Processamento de Imagens (DPI), pela arte de ensinar.

Ao meu primeiro orientador João Argemiro, pelo apoio e incentivo e ao meu segundo orientador Antônio Miguel Monteiro, pelo conhecimento passado e principalmente pela paciência e amizade demonstrada durante esses meses.

Aos colegas da Divisão de Processamento de Imagens e Sensoriamento Remoto, que sempre me receberam de braços abertos. Em especial ao Juan Carlos P. de Garrido, pela amizade e ajuda, principalmente, nos momentos em que nada dava certo e ao Dr. Bernardo F. T. Rudorff que sempre me incentivou do início ao fim.

A meus pais, Luigi e Dalva, pelo amor e incentivo e a minha esposa e filho, Tânia e Daniel Kenzo, sem os quais eu nunca teria chegado até aqui. Impossível expressar em palavras meu amor e gratidão por vocês!

Cada um de vocês contribuiu de uma forma especial para este trabalho.

RESUMO

Instituições governamentais, corporativas e científicas têm realizado grandes investimentos na geração, processamento e distribuição de imagens de sensoriamento remoto. Estes investimentos têm causado um crescimento explosivo nos acervos e bancos de imagens das instituições, superando em muito a atual capacidade de interpretar e analisar estes dados. Paralelo a este crescimento uma nova plataforma tecnológica esta surgindo, os *Web Services*, que são vistos hoje como sendo a próxima onda da evolução da Internet. A visão é de ter uma *Web* rica em funcionalidades como a atual é rica em informações, fato que abre uma oportunidade única para disponibilizar serviços na *Web* que permitam a um usuário qualquer executar processamentos sobre os bancos de imagens, aumentando assim a atual capacidade de interpretar e analisar estes dados, reduzindo custos e agilizando processos de decisão. Com base nesta visão este trabalho propõe uma especificação para um novo serviço geográfico, o WISS – *Web Image Segmentation Service* – que adere às propostas apresentadas pelo OWS *Framework*, em sua versão 0.1.3 apresentada pelo consórcio OGC. É apresentada uma implementação da especificação WISS proposta, que utiliza a biblioteca de componentes geográficos de código aberto TerraLib, e um protótipo demonstrativo foi implementado para o serviço especificado. Por fim, é feita uma discussão sobre o uso das especificações de serviços *Web* no contexto do OGC e suas limitações para aplicações que operam sobre grandes bases de imagens de sensoriamento remoto.

WEB IMAGE SEGMENTATION SERVICE: SPECIFICATION AND IMPLEMENTATION

ABSTRACT

Governmental, corporative and scientific institutions have carried through great investments in the generation, processing and distribution of remote sensing images. This context has caused an explosive growth in the repositories of images of these institutions, surpassing the current capacity to interpret and to analyze these data. Parallel to this growth a new technological platform has appeared, the Web Services which is seen today as being the next wave to the evolution of the Internet. The vision is to open for the provision of available processing services in the Web that would allow any user, anywhere to execute a established procedure over a chosen remote sensing images stored in an repository. That shall increases the current capacity to interpret and to analyze these type of data, reducing costs and speeding decision processes. This work considers a specification for a new geographic service, the WISS - Web Image Segmentation Service - which is compliant with the proposals presented for the OWS Framework, in its version 0.1.3 presented by the OGC trust. An implementation of the specification WISS proposed that uses the open source library of geographic components - TerraLib, and a demonstrative prototype was implemented for the specified service. Finally, a discussion on the use of the specifications of Web Services is made in the context of the OGC and its limitations for applications that operate on large remote sensing image databases.

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

CGI - *Common Gateway Interface*

DCOM - *Distributed Component Object Model*

DETER - Detecção de Desmatamento em Tempo Real

ESRI - *Environmental Systems Research Institute*

GML - Geography Markup Language

GWS - *Geospatial Web Services*

HTTP - *Hiper Text Transfer Protocol*

IIOP - *Internet Inter-ORB Protocol*

INPE – Instituto Nacional de Pesquisas Espaciais

JNI - *Java Native Interface*

NASA - *National Aeronautics and Space Administration*

OGC - *Open Geospatial Consortium*

OpenLS - *OpenGIS Location Services*

OWS - *OpenGIS Web Service*

PDA - *Personal Digital Assistant*

PDI - Processamento Digital de Imagens

RMI - *Remote Method Invocation*

SIG - Sistema de Informação Geográfica

SOAP - *Simple Object Access Protocol*

SPRING – Sistema de Processamento de Informações Georeferenciadas

SVG - *Scalable Vector Graphics*

UDDI - *Universal Description, Discovery and Integration*

UML - *Unified Modeling Language*

XML - *eXtensible Markup Language*

W3C - *World Wide Web Consortium*

WCS - *Web Coverage Service*

WFS - *Web Feature Service*

WICS - *Web Image Classification Service*

WISS - *Web Image Segmentation Service*

WMS - *Web Map Server*

WPS - *Web Processing Service*

WSDL - *Web Service Definition Language*

CAPÍTULO 1

INTRODUÇÃO

1.1 Motivação

Instituições governamentais, corporativas e científicas têm realizado grandes investimentos na geração, processamento e distribuição de imagens de sensoriamento remoto. Estes investimentos têm causado um crescimento explosivo nos acervos e bancos de imagens das instituições, superando em muito a atual capacidade de interpretar e analisar estes dados. O Instituto Nacional de Pesquisas Espaciais (INPE), por exemplo, possui um acervo de 30 anos de imagens de sensoriamento remoto que está sendo armazenado em seu Centro de Dados para acesso on-line, totalizando cerca de 130 Tb em dados. Existe ainda uma forte demanda econômica, política, social e governamental por resultados em tempo hábil, oriundos de informações estratégicas destas imagens (Silva, 2004).

Existem outros grandes bancos de imagens disponíveis para acesso on-line, por exemplo, o projeto da *National Aeronautics and Space Administration* (NASA) e *Stennis Space Center* (NASA, 2004) que fornece mosaicos de imagens do satélite Landsat5 e Landsat7 com abrangência mundial.

Outro exemplo interessante é o *WMS Global Mosaic* (NASA, 2005), um mosaico de imagens Landsat com abrangência mundial, produzido com mais de 8200 cenas do Landsat 7. Cada cena requer mais de 500 Mb de espaço para seu armazenamento. As imagens foram coletadas no período de 1999 a 2003 com resolução de 15 metros no pancromático, 30 metros nas 6 bandas do visível e do infra vermelho e 60 metros nas 2 bandas do infra vermelho termal do sensor Landsat7 ETM+. O banco de imagens do site *WMS Global Mosaic* é disponibilizado através de uma implementação para o serviço *WMS - Web Map Service* (Beaujardiere, 2004).

O projeto de Monitoramento Sistemático do Desflorestamento da Amazônia (PRODES) e o projeto de Detecção de Desmatamento em Tempo Real (DETER) são dois sistemas

em operação, que dão apoio à gestão ambiental no plano do governo federal brasileiro, que usam grandes bases de imagens em auxilio as necessidades em dar resposta e sustentação ao desenho de políticas públicas para a região. Ambos os projetos utilizam o processamento digital de imagens (PDI) de sensoriamento remoto como parte de sua metodologia, para mapear áreas desmatadas ou desflorestadas.

Desde 1989, o INPE vem produzindo estimativas anuais das taxas de desflorestamento da Amazônia Legal. A partir do ano de 2003, estas estimativas envolvem classificação e segmentação digital de imagens como uma de suas principais fases.

A metodologia do projeto PRODES (Valeriano et al, 2004), por exemplo, é baseada na interpretação de 215 imagens do satélite Landsat, e consiste, após a etapa de seleção destas imagens e sua disponibilização em um banco de dados geográfico, em uma etapa subseqüente de classificação. Nesta etapa, uma série de processamentos são executados. A primeira fase neste processo é a transformação dos dados radiométricos das imagens com o uso do modelo linear de mistura espectral, visando estimar a proporção dos componentes solo, vegetação e sombra, para cada pixel, a partir da resposta espectral nas diversas bandas do sensor TM, gerando as imagens-fração solo, vegetação e sombra (Shimabukuro e Smith, 1991). Após este processo as imagens-fração são segmentadas, utilizando um algoritmo de segmentação por crescimento de regiões (Bins et al., 1996). Uma vez realizada a segmentação nas imagens derivadas do modelo linear de mistura espectral, o próximo passo é a classificação não supervisionada e o mapeamento das classes não-supervisionadas em classes informativas (desmatamento do ano, floresta, etc.) (Bins et al., 1993). E por fim é realizada a edição do resultado do mapeamento de classes e a elaboração de mosaicos das cartas temáticas de cada Unidade Federativa. Hoje o INPE fornece ao governo e à sociedade brasileira todas as imagens do satélite Landsat utilizadas na interpretação e a informação temática produzida pelo PRODES digital sobre a localização e extensão dos eventos de desmatamento na Amazônia Legal através da divulgação dos resultados pela Internet.

Neste contexto, onde estão disponíveis na Internet as imagens de sensoriamento remoto e os resultados gerados pela interpretação das mesmas é natural, que demandas como a

possibilidade de repetir o procedimento de classificação sobre uma ou mais imagens ou uma área selecionada daquele repositório, ou a possibilidade de utilizar este mesmo procedimento sobre outras imagens disponíveis em outros repositórios ou fornecidas pelo cliente do serviço, comecem a aparecer. Para atender a estas demandas é preciso encontrar uma plataforma tecnológica que ao mesmo tempo possua capacidade de comunicação, por onde possam trafegar dados espaciais com eficiência sob protocolos padrões e abertos, e seja flexível para poder ser acoplada com facilidade aos sistemas e processos que demandam ou geram dados e informações existentes, ou em desenvolvimento, nas organizações.

Dentre os atuais paradigmas de sistemas de informação a plataforma *Web* (W3C, 1994) parece ser a que mais se aproxima destes requisitos. Isto porque a *Web* confere aos sistemas de informação que a utilizam como plataforma de comunicação, os benefícios da flexibilidade, maior acessibilidade e menores riscos de obsolescência e isolamento (Anderson, 2003).

Até há pouco tempo atrás, os típicos Sistemas de Informações Geográficas (SIG), era caracterizado como sendo um sistema capaz de facilitar diversas funcionalidades para a manipulação de dados georeferenciados, com poucas opções para dados de imagens. Em 1996, o SPRING (Câmara et al, 1996) era o único SIG que incluía no mesmo pacote funcionalidades tradicionais de SIG e um módulo dedicado ao tratamento de imagens de sensoriamento remoto. Mais recentemente, já vimos um movimento de integração entre as plataformas típicas do processamento SIG tradicional e as plataformas de PDI (ESRI, 1998), (Intergraph, 2006), (PCI, 2006). Com a ampliação do uso da Internet e com o seu conseqüente desenvolvimento, os SIG deixaram de ser sistemas monolíticos, fechados e centralizados.

O desenvolvimento da tecnologia SIG vem percorrendo sua trajetória. Dos primeiros sistemas com base em *mainframe* e os sistemas com base em computador pessoal para as atuais propostas e algumas implementações de sistemas baseados em Serviços Geográficos Distribuídos. Nesta nova plataforma, a arquitetura possibilita a ligação e as interações simultâneas entre múltiplos servidores e sistemas de tipo diversificado, sem

as tradicionais restrições de sistemas cliente/servidor. O conceito de Serviços Geográficos Distribuídos é baseado na definição de uma aplicação SIG centrada na comunicação dinâmica e distribuída de serviços através de uma rede como meio principal de acesso a dados, disseminação de informação espacial e condução de análises baseadas em SIG (Peng e Tsou, 2003).

Nesta arquitetura, é permitido a uma variedade de clientes o acesso remoto a dados geo-espaciais e a ferramentas de processamento alojadas em servidores. Estes clientes podem ser um especialista qualquer no domínio do problema operando em um computador pessoal, um computador portátil, um *Personal Digital Assistant* (PDA) ou mesmo um telefone celular.

Adicionalmente às funcionalidades disponíveis nos SIG tradicionais, estes sistemas se beneficiam da utilização da *Web* como plataforma de comunicação, utilizando seus variados protocolos de comunicação (W3C, 1994). Em 1994, o *Open Geospatial Consortium* (OGC, 2005) foi criado. Hoje conta com mais de 250 participantes entre companhias, agências governamentais e universidades, e foi criado para promover o estudo e proposição de especificações que facilitem a interoperabilidade entre sistemas envolvendo informação espacial e de localização (Gardels, 1996) (Percivall, 2003). Os produtos do trabalho do OGC são apresentados sob a forma de especificações de interfaces e padrões de intercâmbio de dados.

Há inúmeras iniciativas baseadas nas especificações do OGC, como (Boucelma, 2002) (Essid, 2004) (Deng e Zhao, 2004), que possibilitam o acesso a dados geográficos na forma de mapas digitais, os chamados GWS - *Geospatial Web Services*.

Com essa tecnologia é possível acessar bancos de dados que estejam em qualquer servidor de dados da Internet sem a necessidade de *download* dos arquivos. Um programa de computador, rodando na Internet ou no computador do usuário, estabelece o acesso aos vários serviços de fornecimento de dados, *Web Services*, integrando em um mesmo mapa digital temas armazenados de forma descentralizada.

Neste cenário, a motivação para esta tese nasce do crescente volume de fontes independentes de banco de imagens de sensoriamento remoto disponíveis, fato que abre uma oportunidade única para disponibilizar serviços na *Web* que permitam a um usuário qualquer, para além de visualizar e/ou descarregar os dados para base local, executar operações sobre as imagens destes repositórios, como por exemplo, o algoritmo de segmentação utilizado pelo PRODES. Esta dissertação desenvolve uma especificação para um serviço geográfico na *Web*, que atende a uma das fases da classificação automática de imagens de sensoriamento remoto, a segmentação. Uma especificação da interface de serviço *Web* para Segmentação de Imagem, denominada *Web Image Segmentation Service* - WISS, baseada no arcabouço de serviços proposto pelo OGC, chamado de *OpenGIS Services Framework* (Percivall, 2003) é proposta. Uma implementação desta especificação baseada em uma biblioteca de código fonte aberto TerraLib é desenvolvida (Câmara et al., 2000).

Com essa especificação e sua implementação, um protótipo envolvendo a provisão deste serviço foi construído. Este protótipo auxilia uma discussão sobre as limitações do arcabouço de serviços proposto pelo OGC para as necessidades dos serviços *Web* para o tratamento digital de imagens de sensoriamento remoto em grandes repositórios distribuídos, em contextos operacionais como aqueles dos programas PRODES ou DETER (Shimabukuro et al, 2005).

1.2 Objetivo

Esta dissertação propõe um serviço para processamento de imagens de sensoriamento remoto, especificando uma interface de Serviço *Web* para Segmentação de Imagem - *Web Image Segmentation Service* (WISS), baseado no arcabouço OGC para serviços, chamado de *OpenGIS Services Framework* (Percivall, 2003). Este arcabouço especifica o escopo, objetivos e comportamento de uma série de componentes.

Porém, convém salientar que os serviços originalmente especificados pelo OGC não seguem as recomendações do W3C (W3C, 1994) para definição de serviços *Web*, como *Simple Object Access Protocol* (SOAP) para intercâmbio de dados, *Web Services Description Language* (WSDL) para descrição dos serviços e *Universal Description*,

Discovery, and Integration (UDDI) para registro dos serviços. Apenas, mais recentemente, o trabalho conhecido como *OpenGIS Web Service - OWS 2 Common Architecture: WSDL SOAP UDDI* (Sonnet, 2005) realizou inumeros testes para adicionar o suporte SOAP/WSDL/UDDI as interfaces dos OWS existentes atualmente.

O serviço especificado nesta dissertação foi composto por operações que permitem: 1) Fornecer informações descritivas das habilidades implementadas no serviço; 2) Fornecer informações descritivas do algoritmo de segmentação disponível e como utilizá-lo; 3) Fornecer informações detalhadas do algoritmo, tipos de dados suportados e os parâmetros específicos solicitados; 4) Permitir a execução do algoritmo sobre imagens escolhidas; 5) Apresentar como resposta a espacialização dos segmentos definidos pelo algoritmo de segmentação após a sua execução e as informações descritivas associadas a estes segmentos.

A especificação do Serviço *Web* para Segmentação de Imagens - WISS seguiu as especificações definidas pelo OGC, em seu OWS (Percivall, 2003). Observando o estágio atual da especificação e observando a existência de outros serviços fornecidos com base na especificação definida pelo OGC, permitindo uma avaliação de suas capacidades e limitações em um contexto operacional.

O serviço WISS disponibiliza quatro operações que podem ser solicitadas pelos seus clientes e executadas pelo WISS. Essas operações são:

1) *GetCapabilities*: esta operação permite ao cliente solicitar e receber de volta os metadados do serviço descrevendo as habilidades implementadas no serviço. Este documento inclui uma breve descrição do segmentador disponível e de como utilizá-lo;

2) *GetDescribe*: implementação obrigatória, esta operação permite ao cliente solicitar informações detalhadas do segmentador disponível, o servidor responde a esta solicitação com um documento em *eXtensible Markup Language* (XML) descrevendo em detalhes o segmentador implementado no serviço. Essas informações incluem uma descrição teórica do algoritmo do segmentador, tipos

de dados suportados, os parâmetros específicos solicitados pelo segmentador e os valores padrão para esses parâmetros em caso do cliente não enviar os valores;

3) *GetSegmentation*: implementação obrigatória, esta operação permite ao cliente executar o serviço de segmentação, enviando os parâmetros requeridos pelo serviço. O servidor responde a solicitação retornando um XML com o URL do arquivo de saída nos formatos *Geography Markup Language* - GML ou *Shapefile*;

4) *GetFeature*: esta operação permite ao cliente solicitar e receber de volta a feição rotulada no formato GML.

Por fim, um protótipo de sistema com base na especificação WISS foi desenvolvido. Nele foram integrados a biblioteca TerraLib, onde estão disponíveis os algoritmos de segmentação, e o MapServer (*University of Minnesota*, 1996), utilizado como *front-end* para acesso aos serviços disponibilizados e ao banco de imagens. O banco de imagens foi gerado com a utilização do aplicativo TerraView, construído utilizando os componentes da biblioteca TerraLib. O MapServer foi utilizado para receber solicitações de execução do serviço WISS e para a apresentação dos resultados em forma de mapas digitais ou feições geográficas.

1.3 Estrutura da Dissertação

Esta dissertação está dividida em 6 capítulos, sendo o primeiro uma introdução sobre o trabalho, com seus objetivos, motivações e contribuições. O segundo capítulo, TECNOLOGIAS DE PROVISÃO DE SERVIÇOS NA *WEB*: *WEB SERVICES* apresenta alguns conceitos das tecnologias envolvidas neste trabalho, em particular àquelas relacionadas as especificações, protocolos e arquiteturas para provisão de serviços na *Web*. O terceiro capítulo, OWS: ESPECIFICAÇÃO OGC PARA SERVIÇOS GEOGRÁFICOS NA *WEB*, apresenta algumas especificações para provisão de serviços na *Web* disponibilizadas pelo consórcio OGC e uma discussão sobre as especificações propostas pelo OGC e a especificação proposta pelo consórcio

W3C. O quarto capítulo, WISS – *Web Image Segmentation Service*: ESPECIFICAÇÃO apresenta a especificação de um novo serviço *Web* para segmentação de imagens de sensoriamento remoto, principal contribuição desta dissertação. No quinto capítulo, WISS - *Web Image Segmentation Service*: IMPLEMENTAÇÃO, o projeto e a implementação de um protótipo que oferece o serviço WISS com base na especificação proposta são apresentados. Finalmente no sexto capítulo, CONCLUSÕES, esta dissertação discute os limites da proposta com base nos experimentos realizados e apresenta algumas idéias para futuros trabalhos. No Anexo A uma descrição completa da integração da biblioteca TerraLib e o servidor de Mapas MapServer é oferecida. Esta descrição inclui a metodologia utilizada, as funcionalidades desenvolvidas e a aplicação de teste implementada.

CAPÍTULO 2

TECNOLOGIAS DE PROVISÃO DE SERVIÇOS NA WEB: *WEB SERVICES*

2.1 Visão Geral

O *World Wide Web Consortium* (W3C, 1994), é um consórcio de empresas de tecnologia (atualmente cerca de 500 membros) fundada por Tim Berners Lee em 1994 para levar a *Web* para o seu potencial máximo, através do desenvolvimento de protocolos comuns e fóruns abertos que promovem sua evolução e asseguram a sua interoperabilidade. O W3C desenvolve tecnologias, denominadas padrões da *Web* para a criação e a interpretação dos conteúdos para *Web*. Sistemas desenvolvidos segundo esses padrões, podem ser acessados e visualizados por qualquer pessoa ou tecnologia independente de hardware, software ou plataforma utilizada, de maneira rápida e compatível com os novos padrões e tecnologias que possam surgir com a evolução da Internet.

Para alcançar seus objetivos, o W3C possui diversos comitês que estudam as tecnologias existentes para a apresentação de conteúdo na Internet e criam padrões de recomendação para utilização destas tecnologias.

As atividades do W3C são geralmente organizadas em grupos que são responsáveis pelo desenvolvimento de toda documentação técnica. Atualmente todas as atividades relacionadas ao *Web Services* no W3C são coordenadas por 5 grupos (W3C, 2002): *Web Services Addressing Working Group*, *Web Services Choreography Working Group*, *Web Services Description Working Group*, *XML Protocol Working Group* e *XML Schema Patterns for Databinding Working Group*.

Esses grupos de trabalho desenvolvem as especificações técnicas para as aplicações *Web Services*, como o protocolo XML (W3C, 2000), SOAP versão 1.2 (W3C, 2003b), WSDL versão 2.0 (W3C, 2006 a) e a arquitetura dos *Web Services* (W3C, 2004).

Um *Web Service* é um componente, ou unidade lógica de aplicação, acessível através de protocolos de Internet. Como componentes esses serviços possuem uma funcionalidade que pode ser reutilizada sem a preocupação de como é implementada. O modo de acesso é diferente de alguns modelos anteriores, onde os componentes eram acessados através de protocolos específicos, como o DCOM (Microsoft, 1998), RMI (SUN, 1997) ou IIOP (OMG, 1997).

Há algumas especificações e tecnologias definidas para a construção ou utilização de *Web Services*. Essas especificações e tecnologias endereçam os seguintes requisitos para o desenvolvimento baseado em serviços: a) uma forma comum de representar dados em um formato de mensagens comum e extensível; b) uma linguagem de descrição do serviço, comum e extensível; c) um mecanismo para localizar os serviços armazenados em *Web* sites específicos; d) um mecanismo para descobrir os provedores de serviços.

O XML é a escolha natural para o modo de representação dos dados. Muitas especificações utilizam o XML para representação dos dados, assim como os XML *Schemas* para descrever os tipos dos dados.

O *Simple Object Access Protocol* (SOAP) é um protocolo leve para troca de informações. Parte da sua especificação é composta por um conjunto de regras de como utilizar o XML para representar os dados. Outra parte define o formato de mensagens, convenções para representar as chamadas de procedimento remoto (RPC) utilizando o SOAP, e associações ao protocolo *Hiper Text Transfer Protocol* (HTTP).

A *Web Services Description Language* (WSDL) é uma linguagem baseada em XML, com a finalidade de documentar as mensagens. Esse mecanismo padrão facilita a interpretação dos contratos pelos desenvolvedores e ferramentas de desenvolvimento.

Também é necessária uma forma de localização dos *Web Services*. O protocolo *Discovery Protocol* define um formato para o documento *discovery* e um protocolo para devolver esse documento, possibilitando a localização dos serviços em um *Web site* conhecido. No entanto, é comum que não se saiba as *Uniform Resource Locator* (URL) onde os serviços podem ser encontrados. O *Universal Description, Discovery, and*

Integration (UDDI) é um mecanismo para os fornecedores anunciarem a existência de seus serviços, e para os consumidores localizarem os serviços de seu interesse.

Pode-se definir, resumidamente, um *Web Service* como um serviço de software publicado na *Web* através do SOAP, descrito com um arquivo WSDL e registrado em UDDI, conforme ilustrado na Figura 2.1.

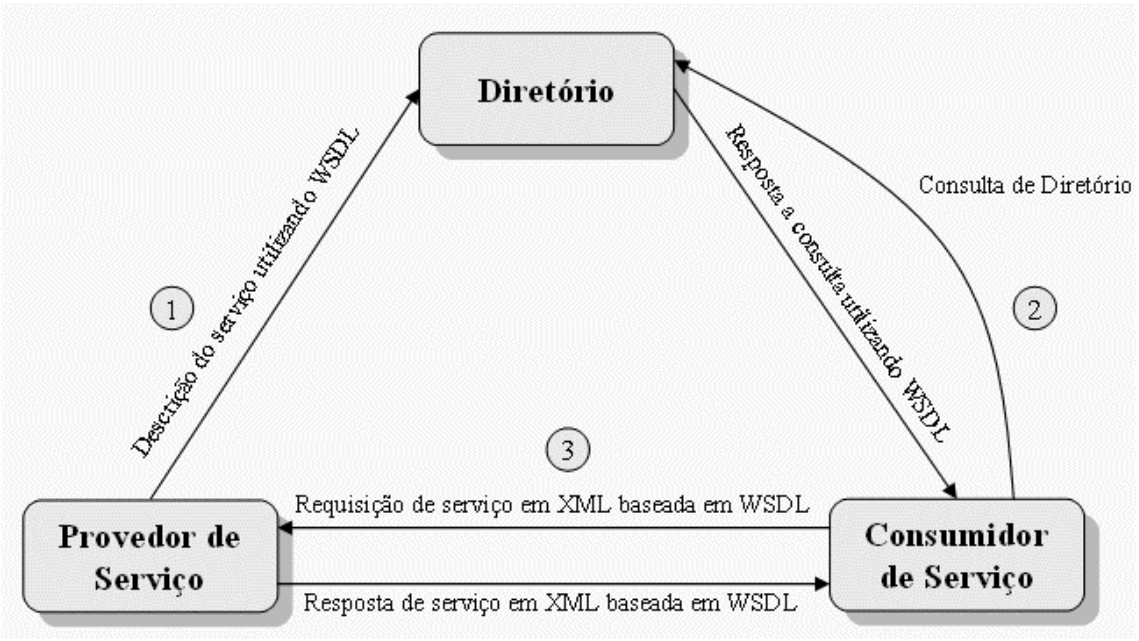

Figura 2.1 – Comunicação via *Web Services*

Fonte: adaptada de W3C (2004)

É importante compreender as tecnologias subjacentes sobre as quais os serviços *Web* são construídos HTTP, XML, SOAP, WSDL e UDDI, os próximos tópicos explicam com mais detalhe cada uma dessas tecnologias. A Figura 2.2 apresenta um diagrama geral da arquitetura em camadas para um *Web Service*.

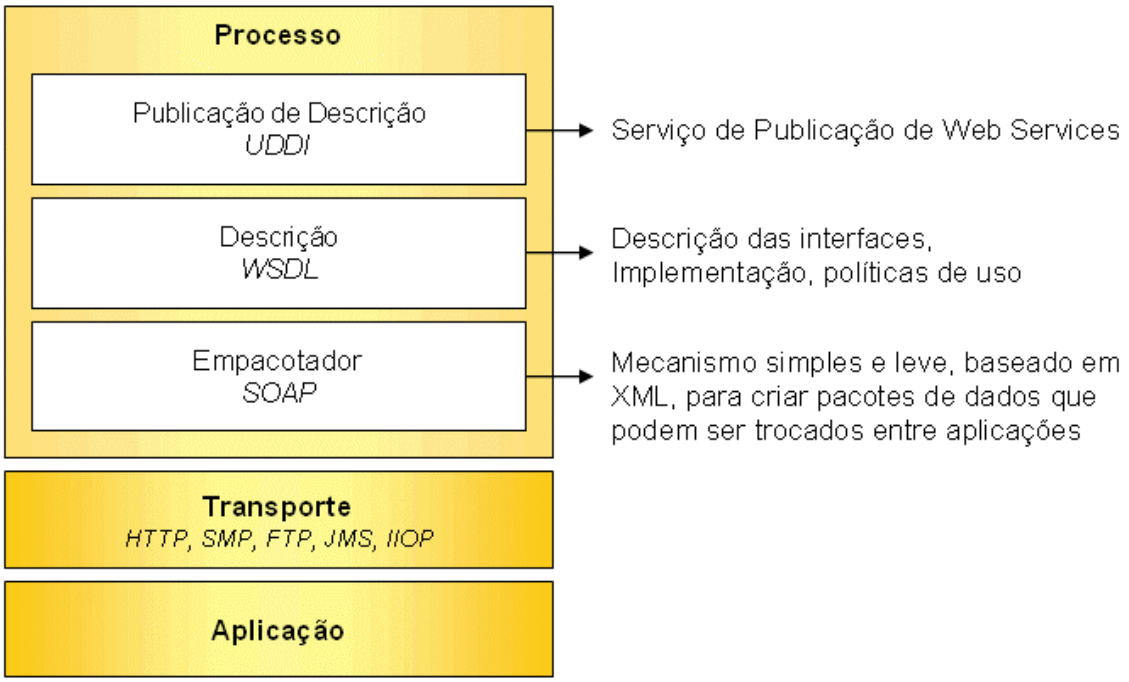

Figura 2.2 –*Web Services* em Camada

Fonte: adaptada de W3C (2004)

2.2 HTTP - *Hiper Text Transfer Protocol*

Os protocolos da Internet são normas acordadas para troca de dados entre redes em diversas plataformas e diferentes ambientes.

HTTP é conhecido como um protocolo *stateless*, isso significa que ele não guarda registros de transações anteriores. Ele também é um dos protocolos mais utilizados na camada processo/aplicação. A utilização do HTTP para comunicação é normalmente dividida em duas partes: uma solicitação feita pelo navegador (ou outro cliente) para um servidor pedindo informações e uma resposta fornecida pelo servidor atendendo a solicitação do cliente.

HTTP utiliza *Uniform Resource Locators* (URL) para auxiliar na localização de documentos em servidores *Web*. Um URL é associado a um endereço *Internet Protocol* (IP) de nível inferior. Normalmente um URL de HTTP, ou seja, o endereço *Web*, possui uma forma similar a:

```
http://www.inpe.br/dsr/index.php
 (1)         (2)     (3)
```

A primeira parte (1) do URL é o protocolo. A segunda parte (2) é o *hostname*, que especifica o domínio ou o servidor que contém os recursos que queremos acessar. A última parte (3) especifica o caminho do documento, que é um conceito similar a localizar arquivos no disco rígido do computador.

A *Internet Corporation for Assigned Names and Numbers*, ou ICAN é o corpo de logística técnica da Internet, criado em outubro de 1998, o ICAN gerencia a atribuição dos identificadores globalmente únicos a se seguir para que a Internet funcione:

- Nomes de domínios da Internet;
- Números de endereço IP;
- Parâmetros e números de portas de protocolos

O mais importante, em nosso contexto, é recordarmos que o HTTP é um protocolo que não guarda registro de transações anteriores (*stateless*).

2.3 XML - *eXtensible Markup Language*

XML é o meio de comunicação de todos os dados e solicitações que entram ou saem de um serviço *Web*.

Esta linguagem possui uma estrutura similar ao do HTML. Apesar de o HTML ser uma ferramenta que foi construída para solucionar um problema específico, XML é uma ferramenta de propósitos gerais que pode ser aplicada a muitos problemas. XML é uma maneira de codificar os dados, sem nenhuma das informações de renderização visual fornecidas pelo HTML.

HTML define *tags* que possuem significados específicos, como *table*, *hr* e *input*. Em XML, você define seus próprios *tags* com significados que suas aplicações possam exigir. XML pode ser considerado um grande conjunto de HTML. Em grande parte, o

HTML segue as regras do XML, então grande parte do HTML seria aceito por um *parser* de XML.

Em essência, XML é uma maneira de comunicar dados estruturados entre aplicações e sistemas de computadores, mesmo quando estão envolvidas múltiplas arquiteturas, linguagens e padrões.

XML é projetado para codificar dados e informações sobre os dados (metadados). Tanto os dados quanto os metadados podem ser qualquer coisa que uma aplicação exija. No caso do HTML, as únicas informações de que o navegador *Web* precisa conhecer são relacionadas a formatação dos dados na tela do usuário. Em XML, é possível incluir informações sobre as dimensões, peso e requisitos elétricos de um produto; sobre a população de um estado ou suas temperaturas médias, mais baixa e mais alta; sobre os dados demográficos de um cliente e seus hábitos de compra; ou sobre qualquer outra coisa que se possa imaginar. É possível armazenar essas informações em HTML, mas seria muito difícil identificar através de programas de computador, qual seria cada parte dos dados.

Um código HTML é consumido por um navegador *Web*, o XML pode ser consumido por um outro código dentro de uma aplicação, por código de uma aplicação separada, por um sistema de gerenciamento de banco de dados ou mesmo exibido ao usuário, formatado através do uso de XSL.

XML é um padrão aberto e é suportado por muitos fornecedores em muitas plataformas. Ele abre as barreiras entre diferentes aplicações, banco de dados, sistemas operacionais e arquiteturas de computadores. Permite que dados sejam processados pela aplicação recipiente em vez de simplesmente serem exibidos ao usuário. Os dados XML são auto-descritivos, incluem tanto os dados quanto as informações sobre os dados. XML é a língua universal dos sistemas de computadores na Internet, contanto que sistemas provedores e consumidores de serviços sigam os esquemas XML e concordem da maneira como os dados devem ser utilizados, quaisquer dos sistemas podem se comunicar.

Muitas tecnologias baseadas em XML podem facilitar ou dar suporte a serviços *Web* geográficos podendo gerar um encadeamento de serviços, podendo um mesmo serviço assumir o papel de provedor ou consumidor. As próximas seções destacam algumas destas tecnologias com uma ênfase em como podem ser usadas e suas limitações, com respeito às exigências da comunidade que trabalha com dados geo-espaciais.

2.4 SOAP - *Simple Object Access Protocol*

SOAP é um padrão para enviar dados de um lado para o outro, entre um cliente e um servidor. Baseia-se em XML e é um protocolo simples e leve. SOAP é um padrão de desenvolvimento e não é de propriedade de nenhum representante específico.

SOAP implementa a funcionalidade de chamada de procedimentos remotos para serviços *Web*. Quando um aplicativo chama um serviço *Web*, um nível de complexidade é adicionado. O código que implementa o serviço pode estar em um nó distante da Internet, sendo executado em processador e sistema operacional incompatíveis. Para fazer isso funcionar, o cliente e o servidor devem implementar um protocolo comum. SOAP é o protocolo padrão projetado para solucionar esse problema. Utilizando XML, SOAP manipula a codificação e a decodificação dos dados estruturados que são enviados e recebidos entre o cliente e o servidor de um serviço *Web*.

SOAP é um protocolo escolhido para o acesso a serviços *Web*. As mensagens SOAP são projetadas para funcionalidade tipo RPC (XML-RPC, 1999) e são métodos de acesso mais ricos em elementos. SOAP é independente do protocolo de transporte. Ele pode viajar em HTTP, SMTP ou *sockets* simples (*raw sockets*, conhecidos como *Direct Internet Messaging* ou DIME). Os serviços *Web* também podem ser acessados através de protocolo HTTP com as mensagens GET e POST.

2.4.1 Estrutura de um Documento SOAP

O W3C define esquemas padrão para mensagens SOAP, utilizando *namespace* XML. Os elementos de uma mensagem SOAP são *Envelopes*, *Header* e *Body*. Abaixo são apresentados cada um desses componentes.

2.4.1.1 Envelope

O *Envelope* é o elemento raiz de todas as mensagens SOAP e é necessário. Similar ao envelope recebido pelo correio, o principal propósito desse elemento é conter a mensagem. A mensagem SOAP a seguir será usada como exemplo, utilizando o esquema XML do *soap-envelope* e chamando-o *namespace env*.

```
<env:Envelope xmlns:env="http://www.w3.org/2001/09/soap-envelope">
   <env:Header>
      <n:boguscontrol xmlns:n="htp://bougs.org/boguscontrol">
         <n:registerno>999999</n:registerno>
      </n:boguscontrol>
   </env:Header>
   <env:Body>
      <m:mymethod xmlns:m="http://bogus.org/mymethod">
         <m:param1>13</m:param1>
         <m:param2>88</m:param2>
      </m:mymethod>
   </env:Body>
</env:Envelope>
```

A primeira e a última linha desse exemplo limitam o *Envelope*. Tudo o que esta entre o *tag* de início <env:Envelope> e o *tag* de final </env:Envelope> é o conteúdo de *Envelope*.

2.4.1.2 Header

O elemento *Header* contém informações relativas a mensagem. O cabeçalho é o mecanismo de capacidade de extensão do SOAP. As informações reais enviadas dentro do cabeçalho não são definidas pelo protocolo SOAP. O desenvolvedor que projeta o serviço *Web* decide quais informações, se houver alguma, devem ser especificadas no cabeçalho. O elemento *Header* é opcional em um documento SOAP. Se for utilizado, tem que ser o primeiro elemento filho do Envelope. O elemento *Header* não contém diretamente as informações do cabeçalho. Em vez disso, ele é um contêiner para os blocos do cabeçalho. Todos os elementos filhos do elemento *Header* são blocos do cabeçalho. O bloco *Header* fornece um mecanismo para especificar as informações que são relevantes a toda a mensagem.

No exemplo apresentado é incluído um único bloco de cabeçalho. O bloco de cabeçalho é o *registerno* (abreviatura de *register number*). Esse serviço *Web* pode por exemplo,

utilizar um modelo *shareware* que exija o envio de um número de registro para que seus métodos sejam chamados.

2.4.1.3 Body

Toda mensagem SOAP inclui um elemento *Body*. O elemento *Body* é o segundo elemento filho do *Envelope* e aparece depois do *Header*. Se nenhum elemento *Header* aparecer em determinada mensagem, o elemento *Body* será o primeiro elemento filho do *Envelope*. Assim como o elemento *Header* serve como contêiner para blocos de cabeçalho, o elemento *Body* serve como contêiner de blocos de corpo.

Os blocos de corpo contêm o conteúdo real da mensagem. No caso de uma chamada de método para um serviço *Web*, o bloco de corpo conteria o nome do método a ser chamado com os valores de seus parâmetros de entrada. A estrutura e o tipo de dados enviados a um bloco de corpo não são definidos pelo padrão SOAP. Entretanto, são especificadas as maneiras através das quais os dados são codificados para inclusão no corpo. O único caso em que a estrutura e o tipo de dados enviados são definidos pelo padrão é o SOAP *Fault*.

Um SOAP *Fault* é um bloco do corpo definido pelo padrão SOAP e é utilizado para comunicar informações de erros. Um elemento *Fault* inclui o número do erro (*faultcode*) e sua fonte (*faultactor*). Um SOAP *Fault* também pode incluir outras informações detalhadas sobre o erro. Uma mensagem SOAP pode conter, no máximo, um elemento *Fault*.

O corpo do exemplo apresentado é uma chamada de método ao método *mymethod*. Dois parâmetros estão sendo passados: param1 com valor 13 e param2 com um valor 88.

O mais relevante em nosso contexto, é recordarmos que o SOAP é independente do protocolo de transporte e as informações contidas no cabeçalho fornecem um mecanismo para os serviços *Web* especificarem um destino final de uma mensagem. Com a possibilidade de serviços gerarem um encadeamento de serviços ou a execução

de um processo demorar muito tempo, é muito importante ter especificado o destino final de uma mensagem.

2.4.2 *Namespace* SOAP

XML não define nenhum *tag* e SOAP é baseado em XML, cada equipe de desenvolvedores precisa chegar a um acordo com relação ao esquema, estrutura, elementos, *tags* e atributos das mensagens XML que a aplicação utilizará para se comunicar. Os códigos que enviam e recebem mensagens SOAP devem conhecer o esquema SOAP. Como definimos um esquema baseado em um modelo totalmente aberto como o XML? Utilizamos *namespace* XML e consultamos os esquemas publicados na Internet pelo W3C.

2.4.3 Utilizando SOAP em serviços *Web*

SOAP é utilizado para implementar chamadas de procedimentos remotos e trocar dados entre código que utiliza um serviço *Web* (o cliente) e o serviço *Web* propriamente dito (o servidor). SOAP e XML são os meios de comunicação entre serviços *Web* e clientes de serviços *Web*.

2.5 WSDL - *Web Services Description Language*

WSDL é um padrão independente de fabricante, para definição de interface de serviços *Web*. Um documento WSDL define todos os métodos expostos pelos serviços *Web*; os nomes, tipos de dados e ordem dos parâmetros; e os tipos de dados retornados. É descrito como o contrato entre o serviço *Web* e os clientes desse serviço *Web*.

WSDL baseia-se em XML e pode ser utilizado para definir a interface de um serviço *Web* que utiliza mecanismos diferentes do SOAP, como HTTP GET/POST.

2.5.1 Estrutura do documento WSDL

Um documento WSDL é uma descrição completa da interface de um serviço *Web* e inclui *Type*, *Message*, *PortType*, *Binding*, *Port* e *Service*. Abaixo são apresentados cada um deles com mais detalhes.

2.5.1.1 Type

WSDL pode utilizar qualquer esquema XML para definir os tipos de dados de parâmetros e valores retornados. O WSDL recomenda o esquema XSD para definir um conjunto de tipos de dados padrão.

Muitos tipos de dados padrão estão incluídos. Alguns exemplos de tipos de dados simples são *booleanos*, que podem aceitar valores de 0 a 1; string, que armazena uma seqüência de caracteres alfanuméricos; *dateTime*, que armazena uma data e hora; *integer*, que armazena um numero inteiro; e o *float* e *double*, que armazenam valores de ponto flutuante. Tipos complexos também são suportados. Um *array* armazena múltiplos valores de tipos de dados simples, que são acessados por um índice. *Struct* armazena múltiplos valores de tipos de dados simples diferentes, que são acessados por nome.

2.5.1.2 Message

Em WSDL, um método é chamado de mensagem. Cada elemento *Message* define um único método de um serviço *Web*. O elemento *Message* define o nome do método. Cada parâmetro de entrada do método é definido por um elemento *Part*, que inclui atributos do nome do parâmetro e do tipo de dados. Todos os elementos *Part* de determinado método são elementos filhos diretos do elemento *Message* correspondente.

Se um método possui um valor de retorno, um segundo elemento especial possui o mesmo atributo nome que o primeiro elemento *Message* do método, com a palavra *Response* anexada. Esse elemento *Message* de resposta possui um único elemento *Part* que descreve o valor de retorno do método.

2.5.1.3 PortType

Um elemento *Message* descreve os parâmetros de entrada de um método ou o valor de retorno de um método. Para tornar a definição da interface explicita, o elemento *PortType* é utilizado. Esse elemento é um container de *tags* de operação. Cada *tag* de

operação define o nome da *Message* de entrada e da *Message* de saída (resposta) de um único método de serviço *Web*.

2.5.1.4 Binding

Os elementos *Binding* especificam o protocolo e a codificação a serem utilizadas para cada método de um serviço *Web*. Cada elemento *Binding* é considerado um ponto de entrada ao serviço *Web*. Todo método exposto pelo serviço *Web* possui seu próprio elemento *Binding*. Se o nome de um método esta sobrecarregado, cada sobrecarga possui em elemento *Binding* separado.

2.5.1.5 Port

Cada elemento *Port* define uma interface completa do serviço *Web*. Ele especifica o arquivo que deve ser utilizado para processar solicitações e o nome do elemento *Binding* que define o protocolo e a codificação para a chamada de métodos.

2.5.1.6 Service

O *tag Service* define o nome do serviço *Web* e inclui *tags* filhos *Port* para cada interface exposta pelo serviço *Web*, podendo expor múltiplas interfaces. Como uma interface é definida por um *tag Port* em WSDL, um serviço *Web* com múltiplas interfaces terá múltiplos *tags Port*. O *tag Service* é simplesmente o proprietário, ou *tag* pai, da coleção de *tags Port*.

O mais importante em nosso contexto, é ressaltar que descrever as interfaces do serviço *Web* não é o bastante. Um serviço *Web* geográfico necessita de um mecanismo para descrever as características dos dados que os vários serviços geográficos podem servir ou processar. O OGC consegue atualmente suprir essa necessidade, disponibilizando para cada serviço *Web* geográfico uma operação chamada *getCapabilities* que retorne, entre outras informações, detalhes sobre os dados ou os tipos de dados que esse serviços dão suporte.

2.6 UDDI - *Universal Description, Discovery and Integration*

UDDI é um padrão de um banco de dados distribuído na Internet e seus serviços *Web*. O próprio UDDI pode ser considerado um serviço *Web*, aplicações acessam UDDI através de mensagens SOAP.

UDDI permite que serviços anunciem e as aplicações descubram serviços *Web* na Internet. Utilizar UDDI para dar *download* no documento WSDL de um serviço *Web* pode ser descrito com uma forma de vinculação precoce, o desenvolvedor consulta o documento WSDL em busca de informações de interface durante a codificação e registra essas informações na memória *cache* para o uso por sua aplicação durante a execução.

2.6.1 Estrutura

UDDI armazena informações sobre serviços *Web* em um esquema XML. Esse esquema inclui os seguintes tipos de informações:

- Informações de Negócios: O elemento *businessEntity* armazena o nome de empresas que fornecem os serviços *Web*. Pode conter também endereço, contato e informações descritivas sobre a empresa. Todas essas informações são chamadas de dados das páginas brancas. Esse elemento também pode armazenar informações de categoria sobre uma empresa. Informações de categoria chamam-se de dados de páginas amarelas. Esse elemento é o elemento raiz do documento XML e possui elementos filhos *Service information*.

- *Service information*: Os elementos *businessService* são utilizados para categorizar serviços *Web* em agrupamentos funcionais. Por exemplo, o desenvolvedor pode requerer todos os serviços *Web* relacionados ao processo de aprovação de propostas de sua organização. Cada elemento *businessService* representa um grupo de serviços *Web*. Esse elemento é um filho direto do elemento *businessEntity*. O elemento *businessService* possui elementos filhos

Service Specification, que será descrito em breve. O *Service Information*, com o *Binding Information*, é chamado de dados das páginas verdes.

- *Binding Information*: O elemento *bindingTemplate* contém informações técnicas sobre os serviços *Web* reais, incluindo os endereços e as informações de roteamento dos serviços *Web*. As informações sobre os padrões e especificações suportadas pelo serviço *Web* também são incluídas aqui. Esse elemento é um filho do elemento *businessService*. As informações de vinculação são chamadas de dados das páginas verdes. Esse elemento contém referências aos elementos *Service Specification*, descrito a seguir.

- *Service Specifications*: O elemento *Model* é um ponteiro para a especificação de interface de um serviço *Web*. Esse elemento não descreve a interface diretamente. Em vez disso, ele inclui o URL da especificação da interface. Isso permite que múltiplos serviços e organizações compartilhem uma interface publicada e desenvolvam serviços *Web* que possam aderir a essa interface.

O esquema UDDI vai além de simplesmente descrever uma interface. UDDI fornece informações sobre que servidores fornecem o serviço *Web*, permite que serviços sejam agrupados e permite também que quem publica o serviço *Web* forneça suas informações de contato.

O principal obstáculo para adoção deste serviço pela comunidade de usuários de dados geo-espaciais é que os registros de UDDI não suportam atualmente nenhum tipo de consulta espacial nos seus catálogos de serviços publicados. Não poder informar a área geográfica desejada para procurar por serviços ou por dados constitui uma limitação real para os usuários. Espera-se que as versões futuras do UDDI terão suporte para tal funcionalidade.

2.7 Revisão

O objetivo desta seção foi fazer uma revisão do que é relevante em cada item apresentado (HTTP, XML, SOAP, WSDL e UDDI) para o WISS e que pode ter impactos na efetivação do serviço.

Em nosso contexto o protocolo HTTP por ser *stateless*, isto é, não guardar registros de transações anteriores, limita o tempo de processamento para grandes imagens de sensoriamento remoto. O período de conexão (*time-out*) com um servidor *Web* normalmente expira antes de 90 segundos, este tempo é muito pequeno quando estamos trabalhando com uma imagem que pode ter facilmente 108 milhões de *pixels*, considerando 3 bandas com 6000x6000 *pixels*.

SOAP por ser independente do protocolo de transporte e por permitir inserir informações no seu cabeçalho que permitam especificar o destino final de uma mensagem de serviço *Web*, pode ser uma boa possibilidade para contornar as limitações do protocolo HTTP, permitindo informar ao consumidor do serviço *Web* o término do processamento solicitado.

Em nosso contexto o WSDL necessita de um mecanismo para descrever as características dos dados que os vários serviços geográficos podem servir ou processar. O OGC consegue atualmente suprir essa necessidade, disponibilizando para cada serviço *Web* geográfico uma operação chamada *getCapabilities* que retorne, entre outras informações, detalhes sobre os dados ou os tipos de dados suportados por esses serviços.

O principal obstáculo para adoção do UDDI é por não suportar atualmente nenhum tipo de consulta espacial nos seus catálogos de serviços publicados. As especificações do OGC para serviços geográficos também não oferecem suporte a consultas espaciais (restrições por área ou geometria), para suprir essa necessidade existe a especificação *OpenGIS Catalog Services* (Kottman, 1999) que introduz um serviço para a publicação e busca em coleções de informações descritivas (metadados) de dados espaciais e

objetos relacionados, com um conjunto de metadados bem estruturados é possível suprir essa necessidade.

CAPÍTULO 3

OWS: ESPECIFICAÇÃO OGC PARA SERVIÇOS GEOGRÁFICOS NA *WEB*

3.1 Visão Geral

O *Open Geospatial Consortium* (OGC, 2005) é um consórcio com mais de 250 companhias, agências governamentais e universidades, criado para promover o desenvolvimento de tecnologias que facilitem a interoperabilidade entre sistemas envolvendo informação espacial e localização (Gardels, 1996) (Percivall, 2003). Os produtos do trabalho do OGC são apresentados sob forma de especificações de interfaces e padrões de intercâmbio de dados.

3.2 Especificações OGC

As especificações do OGC baseiam-se em um *framework* arquitetural (Figura 3.1), chamado de *OpenGIS Services Framework*, onde estão especificados o escopo, objetivos e comportamento de uma série de componentes (Percivall, 2003). Este *framework* pode ser agrupado em três categorias:

Serviços de Dados: componentes que oferecem os serviços básicos de acesso e visualização aos dados geográficos. Exemplos desses serviços incluem o *Web Map Service* (WMS) que produz mapas em duas dimensões visuais, *Web Coverage Service* (WCS) produz acesso as informações espaciais não *renderizadas* onde é necessário uma aplicação cliente para *renderizar* as informações, o *Web Feature Service* (WFS) permite recuperar os dados espaciais no formato *Geography Markup Language* (GML).

Serviços de Processamento: componentes que oferecem os serviços de processamento e transformação de dados recebendo parâmetros específicos para cada processo, desta forma podem fornecer funções de processamento genéricas tais como: projeção e conversão de coordenadas, manipulação de imagens, ou classificação de imagens. Exemplos desses serviços incluem o *Web Coordinate Transformation Service* (WCTS), *Web Processing Service* (WPS) e *Web Image Classification Service* (WICS).

Serviços de Registro ou Catálogo: componentes que oferecem os serviços aos usuários ou aplicações para classificar, registrar, descrever, consultar, manter e acessar informações referentes aos *Web Services*. Incluem o *Web Registry Service* (WRS) e *OpenGIS Catalog Service*.

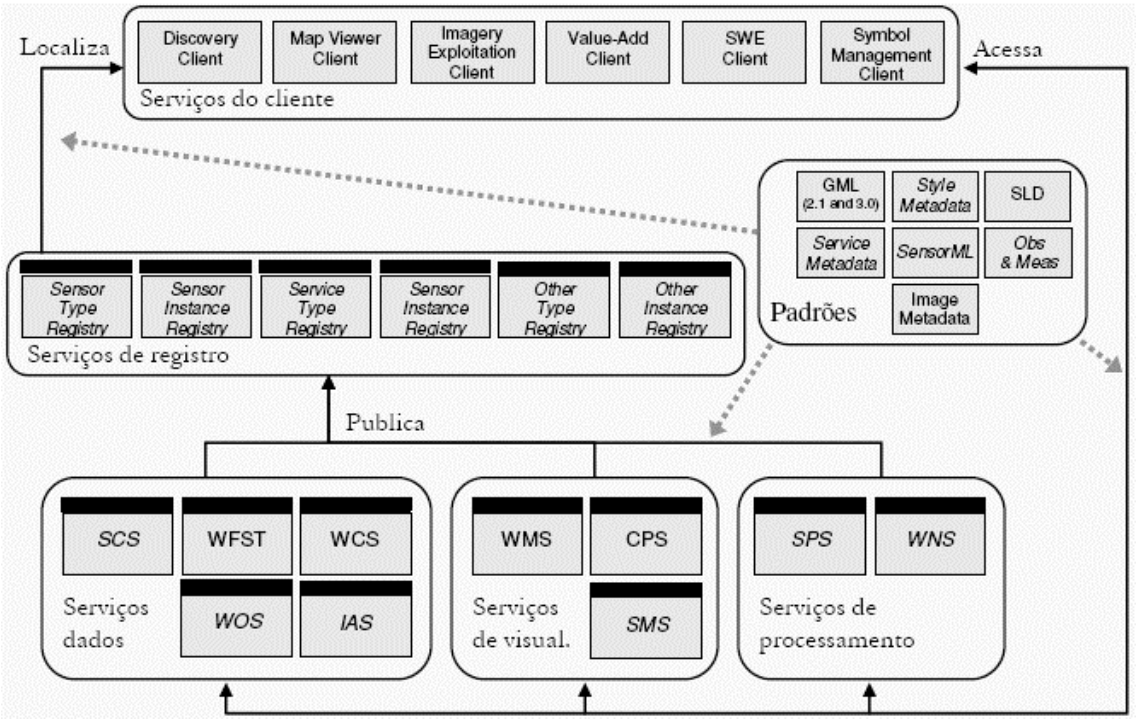

Figura 3.1 - Componentes do OGC *Web Service Framework*
Fonte: adaptada de Percivall (2003) citado por Casanova et al (2005, cap11).

O restante deste capítulo resume alguns serviços definidos pelo OGC. Para maiores detalhes, recomenda-se uma visita ao *Web* site do OGC (OGC, 2005).

3.2.1 Geography Markup Language

A linguagem *Geography Markup Language* (GML) foi criada pelo OGC com o intuito de permitir o transporte e armazenamento de dados geográficos, incluindo propriedades espaciais e não espaciais de feições geográficas (Cox, 2003).

GML encontra-se atualmente na versão 3.1.0, tendo como objetivo oferecer um conjunto de regras com as quais um usuário pode definir sua própria linguagem para descrever seus dados. Para tanto, a GML é baseada em esquemas XML (XML *Schema*). O esquema XML define os elementos (*tags*) usados em um documento que descreve os dados. Sua versão 3.1.1 inclui esquemas que contêm os modelos de geometria, feições, superfícies, sistemas de coordenadas de referência, topologia, informação temporal e feições dinâmicas, unidades de medida, metadados e grid. Todos os esquemas estão publicados nas especificações do OGC (Cox, 2003).

Um estudo da GML e suas limitações na representação da superfície no espaço e tempo simultaneamente esta disponível em (Busquim, 2003).

3.2.2 Web Map Server

A especificação *OpenGIS Web Map Service* (WMS) define um serviço para a produção de mapas na Internet. Neste sentido, o mapa é uma representação visual dos dados geográficos, um desenho, e não os dados de fato. Os mapas produzidos são representações geradas em formatos de imagem, como PNG (W3C, 1995) e JPEG ou em formatos vetoriais, como o *Scalable Vector Graphics* (SVG) (W3C, 2006 b).

Quando o cliente requisita um mapa utilizando o serviço, um conjunto de parâmetros deve ser informado ao servidor: as camadas desejadas, os estilos que devem ser aplicados sobre as camadas, a área de cobertura do mapa, a projeção ou sistema de coordenadas geográficas, o formato da imagem gerada e também o seu tamanho.

O serviço possui as seguintes operações:

- *GetCapabilities*: obtém os metadados do servidor, que descrevem o conteúdo e os parâmetros aceitos.
- *GetMap*: obtém a imagem do mapa que corresponde aos parâmetros informados.
- *GetFeatureInfo*: recupera informações sobre um elemento (*feature*) particular de um mapa.

A especificação WMS para a arquitetura apresentada na Figura 3.2, conta com três componentes: o cliente WMS, o servidor de mapas WMS e a base de dados. O cliente WMS é um programa que processa as requisições do usuário e visualiza dados espaciais. É uma série de páginas HTML geradas dinamicamente e executadas dentro do navegador, as quais se comunicam diretamente com um servidor de mapas via um protocolo HTTP através de três interfaces de requisição WMS, *GetCapabilities*, *GetMap*, e *GetFeatureInfo.*

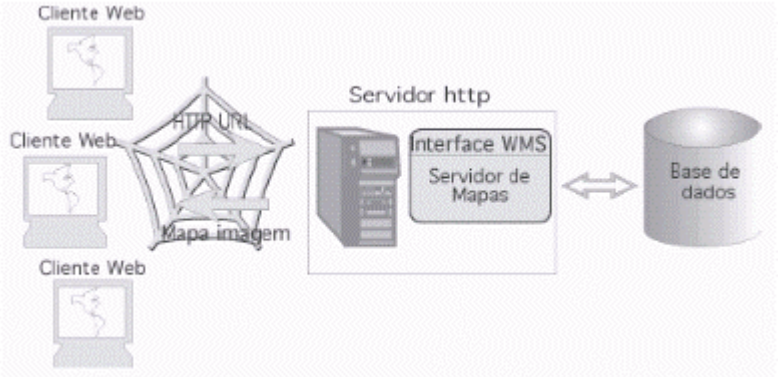

Figura 3.2 - Configuração básica da especificação WMS

Fonte: Miranda (2003).

A figura acima representa uma interação típica cliente e servidor WMS que inicia com o cliente WMS requisitando um *GetCapabilities* do servidor de mapas WMS de maneira a determinar o que ele pode fazer e quais mapas ele pode oferecer. Em seguida, o cliente WMS requisita um mapa imagem de acordo com estas capacidades, usando o *GetMap*. Finalmente, o cliente WMS pode escolher um ponto no mapa imagem e requisitar mais atributos acerca deste ponto, através do *GetFeatureInfo.*

Em resposta a uma requisição *GetCapabilities*, o servidor WMS produz um documento XML contendo os serviços disponíveis, descrevendo todas as operações permitidas e informações sobre os mapas disponíveis. A aplicação cliente WMS tem que analisar o documento XML para recuperar as informações necessárias para requisição dos mapas. Com estas informações, o cliente WMS pode requisitar um mapa imagem do servidor WMS usando a operação *GetMap*. Em seguida, o servidor WMS processa a requisição, normalmente acessando dados armazenados em uma base de dados. Por fim, o servidor

WMS retorna ao cliente WMS um mapa imagem codificado na forma de uma figura MIME, como GIF ou JPEG. Com o mapa no navegador, o cliente WMS pode requisitar dados de atributos do mapa escolhendo um ponto no mapa, usando a função *GetFeatureInfo*. A resposta do servidor WMS pode vir nos seguintes formatos: arquivo GML, arquivo texto ou arquivo HTML. O cliente WMS analisa o GML e mostra os atributos requisitados do mapa no navegador.

3.2.3 Web Feature Service

A especificação *OpenGIS Web Feature Service* (WFS) define um serviço para que clientes possam recuperar objetos (*features*) espaciais em formato GML de servidores WFS, permitindo a execução de consultas, inserções, atualizações e exclusões de feições geográficas. Para tanto, foram definidas as seguintes operações:

- *GetCapabilities*: retorna um documento que descreve os tipos de feições suportados, bem como as operações permitidas por cada um destes tipos.
- *DercribeFeatureType*: retorna uma descrição de um determinado tipo de feição, suportado pelo WFS.
- *GetFeature*: retorna instância de dados oferecidos pelo serviço, representadas por documentos no formato GML.

Além destas, que definem um WFS básico, que é somente-leitura, existem ainda as seguintes operações, que definem um WFS transacional:

- *Transaction*: permitem a realização de operações de inserção, atualização e exclusão sobre os dados mantidos sob o WFS.
- *LockFeature*: operação opcional, utilizada para bloquear um ou mais itens de dados na utilização de um WFS transacional, permitindo a *serialização*.

A seqüência natural na utilização de *Web Services* que implementam a especificação WFS consiste de três passos. O primeiro é a utilização do método *GetCapabilities* pelo

cliente, que retorna as capacidades do serviço. Em seguida, o cliente pode invocar o método *DescribeFeatureType* e ter detalhes de um determinado tipo de feição. O terceiro passo consiste de recuperar ou modificar dados, utilizando-se o método *GetFeature* ou *Transaction* (Vretanos, 2005).

3.2.4 Web Coverage Service

A especificação *OpenGIS Web Coverage Service* (WCS) tal como definida, é um serviço que disponibiliza dados espaciais existentes sob a forma de coberturas multi-dimensionais. Estas são compostas por valores ou propriedades referentes a localizações geográficas espaçadas de forma regular através de um, dois ou três eixos de um sistema de coordenadas geográfico, podendo também conter informação temporal, regular ou irregularmente espaçada.

Três operações são implementadas no serviço:

- *GetCapabilities*: fornece uma descrição do servidor e informações básicas acerca das coleções de dados disponíveis.
- *DescribeCoverage*: recupera uma descrição completa das *coverages*.
- *GetCoverage*: recupera uma *coverage* (valores e propriedades de um conjunto de localizações geográficas) no servidor.

3.2.5 OpenGIS Location Services

A especificação *OpenGIS Location Services* (Mabrouk, 2005) foi aprovada pelo OGC em janeiro de 2004. Ela define um conjunto de interfaces para o desenvolvimento de serviços baseados em localização, todos utilizando protocolos padrão *Web*. Os serviços especificados encontram-se descritos a seguir:

- **Serviço de Diretório**: provê acesso a um diretório on-line para localização de um determinado lugar, produto ou serviço.

- **Serviço de *Gateway***: identifica a posição geográfica de um determinado dispositivo móvel.

- **Serviço de Geocodificação/Geocodificação reversa**: identifica uma posição geográfica, dado o nome de um lugar ou endereço. Também funciona de forma reversa identificando um endereço completo dada uma posição geográfica.

- **Serviço de Apresentação de Mapas**: apresenta informações geográficas no terminal móvel. É usada para apresentar mapas destacando rotas entre dois pontos, pontos de interesse, área de interesse, localizações e/ou endereços.

- **Serviço de Determinação de Rotas**: determina a rota entre dois pontos informados pelo usuário. O usuário também pode, opcionalmente, informar pontos pelos quais a rota deve passar ou rotas preferenciais (mais rápida, mais curta, menos tráfego, mais atrativa, etc.) e o modo de transporte.

3.2.6 Web Processing Service

A especificação *OpenGIS Web Processing Service* (WPS) define um conjunto de interfaces para executar cálculos e modelos matemáticos em dados geográficos. Os dados geográficos requeridos pelos serviços podem ser disponibilizados localmente ou enviados pela rede no formato padrão *Geography Markup Language* (GML) ou *Geolinked Data Access Service* (GDAS). Os cálculos podem ser desde uma simples subtração de dois dados geográficos a um complicado modelo de mudança climática global.

Esta especificação foi planejada com mecanismos para identificar os dados espaciais requeridos pelo cálculo, iniciar o cálculo e gerenciar os resultados para serem enviados ao cliente. O WPS pode processar tanto dados vetoriais como matriciais. As interfaces especificadas encontram-se a seguir:

- *GetCapabilities*: esta interface permite ao cliente solicitar e receber informações dos serviços disponíveis no servidor, incluindo o nome dos processos e como eles podem ser executados.

- *DescribeProcess*: esta interface permite ao diente adquirir mais informações sobre as operações disponíveis no servidor WPS, incluindo os parâmetros de entrada, formatos e saída dos resultados.

- *Execute*: esta interface permite ao cliente executar um processo específico no servidor WPS com os devidos parâmetros de entrada.

Essas operações têm várias similaridades com outros OGC *Web Services*, incluindo o WMS, WFS e WCS. Algumas dessas interfaces são comuns em outros OWS e são especificadas no *OpenGIS Web Service Common Implementation Specification* (Whiteside, 2005).

3.2.7 Web Image Classification Service

A especificação *OpenGIS Web Image Classification Service* (WICS) define um conjunto de interfaces para executar uma classificação de imagens digitais.

Classificação de imagens digitais é o processo de extração de informação em imagens para reconhecer padrões e objetos homogêneos com o objetivo de mapear as áreas da superfície terrestre. O resultado final de uma classificação é uma imagem temática (mapa), onde os *pixels* classificados são representados por símbolos gráficos ou cores. Cada cor ou símbolo está associado a uma classe (área urbana, tipos de florestas, tipos de solo, etc.) definida pelo usuário (Mascarenhas e Velasco, 1989).

Uma classificação de imagens pode ser feita de duas formas: (a) automática, onde o computador analisa individualmente os atributos numéricos de cada pixel na imagem; (b) não automática, conhecida como foto interpretação onde, um especialista humano extrai as informações baseando-se na inspeção visual da imagem. Essas duas abordagens são importantes e muitas vezes complementares.

A classificação automática pode ser dividida em dois grupos: supervisionada e não supervisionada, as quais são descritas a seguir:

- Classificação não-supervisionada: cada pixel da imagem é associado a uma classe espectral sem que o usuário tenha um conhecimento prévio do número ou identificação das diferentes classes presentes na imagem.

- Classificação Supervisionada: o usuário seleciona amostras de treinamento representativas para cada uma das classes que se deseja identificar na imagem. A partir disso, o classificador, utilizando regras estatísticas pré-estabelecidas, identifica e classificada toda a área em análise para as classes de interesse (Mather, 1999).

As interfaces do WICS suportam as seguintes operações:

- Executar classificação não-supervisionada;

- Executar classificação supervisionada com arquivo de treinamento da classificação disponível no servidor;

- Executar treinamento da classificação supervisionada;

- Fornecer parâmetros para o arquivo de treinamento disponível no servidor;

- Fornecer parâmetros para o treinamento da classificação baseado em dados fornecidos pelo cliente;

- Fornecer suporte ao esquema da classificação.

As interfaces do WICS especificam quatro operações que podem ser solicitadas pelo cliente e executadas no servidor WICS. Essas operações encontram-se a seguir:

- *GetCapabilities*: implementação requerida, esta operação permite ao cliente solicitar e receber informações descrevendo as operações disponíveis no servidor. Este serviço inclui uma breve descrição dos classificadores disponíveis, incluindo a identificação do classificador, tipo de classificador e descrição explicativa de cada classificador;

- *GetClassification*: implementação requerida, esta operação permite ao cliente especificar um classificador e enviar dados (usando um servidor que suporte o tipo e formato do dado enviado), possivelmente juntos com os parâmetros requeridos pelo classificador. O servidor responde a solicitação, classificando os dados usando o classificador identificado pelo cliente, a resposta do servidor pode ser de duas maneiras: retorna o dado classificado ou um endereço/URL do dado classificado.

- *TrainClassifier*: implementação opcional, esta operação permite ao cliente especificar um classificador e enviar os dados de treinamento, juntamente com os parâmetros definidos pelo classificador de acordo com o tipo e formato especificado no servidor. O cliente então precisa identificar um esquema de classificação. O servidor treina o classificador selecionado pelo cliente e retorna os parâmetros do treinamento no formato previamente informado ou uma URL dos parâmetros.

- *DescribeClassifier*: implementação requerida, esta operação permite ao cliente solicitar informações detalhadas dos classificadores disponíveis no servidor. O servidor responde a essa solicitação com documento XML descrevendo os detalhes dos classificadores identificados pelo cliente. Esta descrição pode incluir um breve embasamento teórico dos algoritmos dos classificadores, tipos de dados suportados, parâmetros específicos do classificador e valores padrão para os parâmetros no caso do cliente não enviar os valores solicitados.

Uma característica importante do WICS é que ele deve prover informações detalhadas dos classificadores, deste mesmo modo deve prover informações dos parâmetros dos classificadores e seus esquemas de treinamento, informações estas que são bem diferentes dos metadados para dados geográficos. Por esta razão a operação *DescribeClassifier* responsável por informar os detalhes do servidor é obrigatória.

Um exemplo da implementação do serviço WICS pode ser visto em (Deng, 2006), onde são permitidos aos usuários executarem as operações *getCapabilities*, *getClassification*,

TrainClassifier e *DescribeClassifier*. O usuário pode escolher o algoritmo de classificação (supervisionada e não supervisionada), enviar parâmetros de treinamento para uma classificação supervisionada e executar o processo de classificação em imagens fornecidas pelo usuário.

3.3 W3C e OGC *Web Services*

Os serviços originalmente especificados pelo OGC não seguem as recomendações do W3C para definição de serviços *Web*, como SOAP para intercâmbio de dados, WSDL para descrição dos serviços e UDDI para registro dos serviços. Apenas, mais recentemente, o trabalho conhecido como *OWS 2 Common Architecture: WSDL SOAP UDDI* (Sonnet, 2005) realizou inúmeros testes para adicionar o suporte SOAP/WSDL/UDDI as interfaces dos OWS existentes atualmente.

O *OWS 2 Common Architecture* não é um padrão ou uma especificação, é apenas um relatório que apresenta os resultados e problemas encontrados durante a tentativa de criação de diferentes propostas para adicionar o suporte SOAP/WSDL/UDDI às especificações WMS, WFS, WCS e CS-W.

CAPÍTULO 4

WISS – *WEB IMAGE SEGMENTATION SERVICE*: ESPECIFICAÇÃO

4.1 Visão Geral

A especificação do Serviço *Web* para Segmentação de Imagens (WISS) esta baseada no arcabouço de serviços proposto pelo OGC, chamado de *OpenGIS Services Framework* (Percivall, 2003) e tem como principal função segmentar imagens de sensoriamento remoto.

O processo de segmentação consiste em subdividir uma imagem em regiões homogêneas considerando alguns de seus atributos, como por exemplo, o nível de cinza dos *pixels* e a textura, visando caracterizar a representatividade dos objetos da cena (Bins et al., 1996). A segmentação pressupõe a geração de objetos internamente homogêneos e estatisticamente distintos de seus vizinhos, que serão gerados segundo critérios de descontinuidade ou de similaridade.

O algoritmo de segmentação proposto por (Bins et al., 1996), é baseado na tradicional técnica de crescimento de regiões que consiste em um processo de iteratividade no qual regiões são agrupadas a partir de *pixels* individuais. Tal implementação propõe um agrupamento de regiões baseado no conceito de região vizinha mais similar. Neste método são definidos dois limiares, o primeiro é o de similaridade, abaixo do qual duas regiões são consideradas similares e o segundo é o de área, valor de área mínima para que uma região seja individualizada.

O resultado do processo de segmentação é uma imagem rotulada, cada região apresentando um rótulo (valor de nível digital), que deve ser classificada por classificadores de região (Bins et al., 1993).

O serviço WISS especifica quatro operações que podem ser solicitadas pelo cliente e executadas no servidor WISS. As descrições dessas operações encontram-se a seguir:

- *GetCapabilities*: implementação obrigatória, esta operação permite ao cliente solicitar e receber de volta o metadados do serviço com informações da organização detentora do serviço, sobre as operações especificadas e implementadas no servidor e o resumo do segmentador disponível;

- *GetDescribe*: implementação obrigatória, esta operação permite ao cliente solicitar informações detalhadas do segmentador disponível, o servidor responde a esta solicitação com um documento em XML descrevendo em detalhes o segmentador implementado no serviço. Essas informações incluem uma descrição teórica do algoritmo do segmentador, tipos de dados suportados, os parâmetros específicos solicitados por cada segmentador e os valores padrão para esses parâmetros em caso do cliente não enviar os valores;

- *GetSegmentation*: implementação obrigatória, esta operação permite ao cliente executar o serviço de segmentação, enviando os parâmetros requeridos pelo serviço. O servidor responde a solicitação retornando um XML com o URL do arquivo de saída nos formatos GML ou *Shapefile*. O acesso ao resultado no formato GML do processamento é feito através da operação *GetFeature*;

- *GetFeature*: esta operação permite ao cliente solicitar e receber de volta a feição rotulada no formato *Geography Markup Language* (GML).

Essas operações têm várias similaridades com outros OGC *Web Services*, incluindo o WMS, WFS e WFC. Algumas dessas interfaces são comuns em outros OWS e são especificadas no *OpenGIS Web Service Common Implementation Specification* (Whiteside, 2005).

A Figura 4.1 é uma representação simples da interface WISS. Este diagrama de classes apresenta a interface da classe WISS herdando a operação *getCapabilities* da interface do OGC *Web Services* e adicionando a operação *getDescribe*, *getSegmentation* e *getFeature*. Cada uma das operações do WISS são descritas nos proximos tópicos.

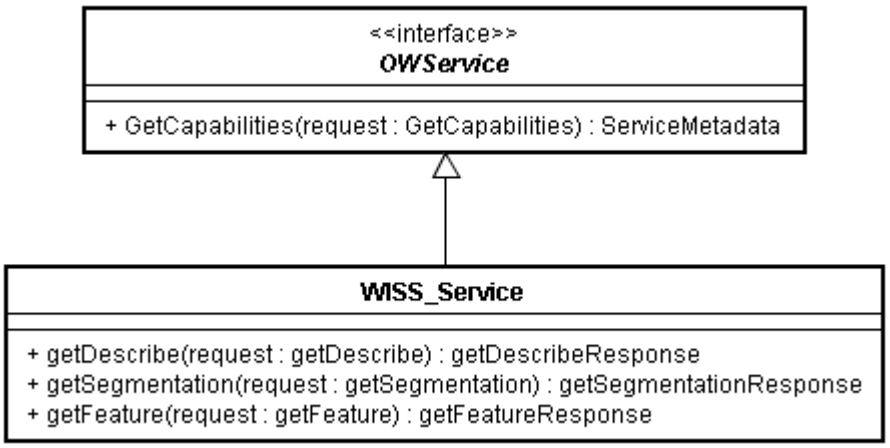

Figura 4.1 – Diagrama UML com a interface WISS

4.2 Operação *GetCapabilities* (requerido)

A implementação da operação *GetCapabilities* é obrigatória, permite aos clientes do serviço WISS recuperar os metadados do serviço com informações da organização detentora do serviço, sobre as operações especificadas e implementadas no servidor e o resumo do segmentador disponível.

4.2.1 Operação *Request*

Os parâmetros para a operação *GetCapabilities Request* é apresentado na Tabela 4.1. Essa tabela também especifica o tipo de dados, definição, valores e obrigatoriedade dos parâmetros.

Tabela 4.1 – *Getcapabilities Request*

Nome	Definição	Tipo	Valores	Multiplicidade e Obrigatoriedade
service	Identifica o tipo de serviço	String	WISS, WMS, WFS, WICS	1 obrigatório
request	Nome da operação	String	*GetCapabilities*	1 obrigatório
version	Identifica a versão do serviço.	String	a.b 0.1	0..1 opcional Quando omitida retorna a versão suportada

4.2.2 Operação *Response*

O documento de metadado completo do serviço contém quatro seções específicas e apresentadas na Tabela 4.2. Dependendo dos valores dos parâmetros da seção da operação *GetCapabilities Request*, algumas combinações dessas seções podem ser retornadas.

Tabela 4.2 – *Getcapabilities Response*

Nome da Seção	Conteudo
ServiceIdentification	Metadado básico sobre o específico servidor. O esquema dessa seção deve ser o mesmo para todos OWS.
ServiceProvider	Metadado da organização detentora do serviço. O esquema dessa seção deve ser o mesmo para todos OWS.
OperationsMetadata	Metadado das operações especificadas e implementadas no servidor, incluindo o URL para operações de *request*.
Contents	Metadado do algoritmo implementado no serviço WISS, contendo o resumo dos dados para a segmentação.

Adicional a esta seção cada documento de metadados retornado pelo serviço deve conter obrigatoriamente a versão (parâmetro '*version*') do serviço.

4.2.2.1 ServiceIdentification

A seção *ServiceIdentification* de um serviço de metadados deve conter um metadado básico sobre o específico servidor. O conteúdo e organização dessa seção devem ser os mesmos para todos os OWS. A seção *ServiceIdentification* deve incluir os parâmetros e elementos apresentados na Tabela 4.3, essa tabela especifica o tipo de dados, a multiplicidade e obrigatoriedade de cada parâmetro listado.

Tabela 4.3 – Seção *Serviceidentification*

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
ServiceType	Nome de um tipo de serviço registrado para o serviço, normalmente utilizado para comunicação de máquina com máquina.	String	1 obrigatório
ServiTypeVersion	Versão do tipo de serviço implementado no servidor.	String	1...N obrigatório
Title	Título do servidor.	String	1 obrigatório
Abstract	Breve descrição do servidor.	String	0...1 opcional
Keywords	Palavra chave usada na descrição do servidor.	String	0...N opcional
Fees	*Fees* é o termo usado para recuperar a origem dos dados.	String	0...1 opcional
AccessConstraints	Garante a propriedade intelectual dos dados informando as restrições para o uso dos dados ou do servidor.	String	0...N opcional

4.2.2.2 ServiceProvider

A seção *ServiceProvider* deve conter um metadado sobre o funcionamento do servidor devendo incluir os parâmetros e elementos apresentados na Tabela 4.4. O conteúdo e organização dessa seção devem ser o mesmo para todos os OWS (Whiteside, 2005).

Tabela 4.4 – Seção *Serviceprovider*

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
ProviderName	Identificador único do provedor (responsável) do serviço	String	1 obrigatório
ProviderSite	Referências mais relevantes do provedor do serviço.	String	0...1 opcional
ServiceContact	Informações de contato com o provedor do serviço.	String	1 obrigatório

4.2.2.3 OperationsMetadata

A seção *OperationsMetadata* de um serviço de metadados deve conter um metadado informando as operações e implementações disponíveis no servidor, incluindo o URL para solicitação dos serviços. O conteúdo e organização dessa seção deve ser o mesmo para todos os OWS, mas serviços individuais podem adicionar outros elementos nesta seção ou mudar os elementos opcionais, tornando-os obrigatório. A seção *OperationsMetadata* deve incluir os parâmetros e elementos apresentados na Tabela 4.5, essa tabela especifica o tipo de dados, a multiplicidade e obrigatoriedade de cada parâmetro listado.

Tabela 4.5 – Seção *Operationsmetadata*

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
Operation	Metadado das interfaces (operações) disponíveis no servidor.	ver tabela 4.6	1...N obrigatório
Parameter	Parâmetros validos para serem aplicados a uma ou mais operações.	ver tabela 4.8	0...N opcional
Constraint	Restrições ou valores válidos para parâmetros enviados ao servidor.	ver tabela 4.8	0...N opcional
ExtendedCapabilities	Metadado sobre o servidor e software adicionais.	String	0...1 opcional

Os conteúdos dos parâmetros *Operation*, *Parameter* e *Constraint* estão incluídos nas Tabelas 4.6 a 4.8. Essas tabelas especificam o tipo de dados e a multiplicidade de cada parâmetro listado.

Tabela 4.6 – Subitem *Operation*

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
Name	Nome da operação de *request*, por exemplo, *GetCapabilities*.	String	1 obrigatório
DCP	Informação para um *Distributed Computing Platform* (DCP).	ver tabela 4.7	1 obrigatório
Parameter	Parâmetros válidos aplicados na operação.	ver tabela 4.8	0...N opcional
Constraint	Restrições ou valores válidos para parâmetros enviados ao servidor	ver tabela 4.8	0...N opcional
Metadata	Metadado sobre a operação.	String	0...N opcional

Tabela 4.7 – Subitem Dcp

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
HTTP	URL para o HTTP DCP.	String	1 obrigatório
Get	Método de *request*.	String	0...N opcional
Post	Método de *request*.	String	0...N opcional

Tabela 4.8 – Subitem Parametros E Constraint

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
Name	Nome ou identificador dos parâmetros.	String	1 obrigatório
Value	Valor válido dos parâmetros.	String	1...N obrigatório
Metadata	Metadado sobre o domínio dos parâmetros.	String	0...N opcional

4.2.2.4 Contents

A seção *Contents* de um serviço de metadados deve conter um metadado informando os parâmetros necessários para executar o serviço. Para o WISS a seção *Contents* deve conter uma breve descrição do processo de segmentação, formatos de imagens

suportados, obrigatoriedade dos parâmetros e seus valores válidos. A seção *Contents* deve conter os parâmetros listados nas Tabelas 4.9 a 4.11.

Tabela 4.9 – Seção *Contents*

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
SegmentationBrief	Descrição do serviço WISS disponível no servidor.	ver tabela 4.10	1 obrigatório

Tabela 4.10 – Subitem *Segmentationbrief*

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
Name	Identificador do segmentador.	String	1 obrigatório
Title	Título para o segmentador.	String	1 opcional
Abstract	Descrição do processo de segmentação.	String	1 opcional
Class	Identificador para a classe do segmentador	String	1 opcional
ParameterSchemaURL	Endereço do documento que descreve os parâmetros de entrada para o segmentador.	String	1 obrigatório
SourceImageFormat	Formato de entrada das imagens suportada pela segmentação.	String	1...N obrigatório
SegCoverageFormat	Formato do resultado de retorno da segmentação.	String	1...N obrigatório
KVPParameter	Definição dos parâmetros, quando usado o método HTTP GET.	ver tabela 4.11	1 obrigatório

Tabela 4.11 – Subitem *Kvparameter*

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
Keyword	Nome do parâmetro.	String	1 obrigatório
ValueType	Tipo do valor correspondente ao parâmetro.	String	1 obrigatório
required	Indica se KVP é obrigatório.	Booleano	1 obrigatório

4.2.3 Exceção

Em um evento no servidor de WISS pode ocorrer um erro no serviço na operação *GetCapabilities Request*, esse evento retornará uma mensagem de exceção. As mensagens de exceção são apresentadas na Tabela 4.12:

Tabela 4.12 – Exceção *Getcapabilities*

Valor	Mensagem	Motivo
MissingParameterValue	*Operation request does not include a parameter value*	Nome do parâmetro não encontrado
InvalidParameterValue	*Operation request contains an invalid parameter value*	Nome do parâmetro com valor invalido.
VersionNegotiationFailed	*"Version" parameter value, in GetCapabilities operation request, did not include any version supported by this server*	Versão não suportada

4.2.4 Exemplos

4.2.4.1 *GetCapabilities Request*

Segue exemplo da operação de *Request* do *GetCapabilities* solicitada por uma URL:

```
http://hostname:port/path?service=WISS&request=GetCapabilities
                          &version=0.1
```

Segue exemplo da operação de *Request* do *GetCapabilities* solicitada por um XML:

```
<?xml version="0.1" encoding="UTF-8"?>
<GetCapabilities xmlns="http://www.opengeospatial.net/ows"
xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/ows
fragmentGetCapabilitiesRequest.xsd" service="WISS">
      <Versions>
            <Version>0.1</Version>
      </Versions>
</GetCapabilities>
```

4.2.4.2 *GetCapabilities Response*

Abaixo estão ilustrados cada seção (*ServiceIdentification*, *ServiceProvider*, *OperationsMetadata* e *Contents*) da operação *GetCapabilities response* no formato XML:

- Seção *ServiceIdentification*

```
<?xml version="1.0" encoding="UTF-8" ?>
<Capabilities     version="0.1"     xmlns="http://www.opengis.net/wiss"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<ows:ServiceIdentification>
  <ows:ServiceType>INPE:WISS</ows:ServiceType>
  <ows:ServiceTypeVersion>0.1</ows:ServiceTypeVersion>
  <ows:Title>INPE_WISS_SERVICE</ows:Title>
  <ows:Abstract>Web Image Segmentation Service at INPE</ows:Abstract>
<ows:Keywords>
  <ows:Keyword>Image Segmentation</ows:Keyword>
  <ows:Keyword>Crescimento de Regioes</ows:Keyword>
  </ows:Keywords>
  <ows:Fees>NONE</ows:Fees>
  <ows:AccessConstraints>NONE</ows:AccessConstraints>
</ows:ServiceIdentification>
```

- Seção *ServiceProvider*

```
<ows:ServiceProvider>
  <ows:ProviderName>DPI, INPE</ows:ProviderName>
<ows:ServiceContact>
  <ows:IndividualName>Luigi C M Aulicino</ows:IndividualName>
  <ows:PositionName>Student</ows:PositionName>
<ows:ContactInfo>
<ows:Phone>
  <ows:Voice>55 12 3945 6510</ows:Voice>
  </ows:Phone>
<ows:Address>
  <ows:DeliveryPoint>Av.     dos     Astronautas,1758     Jardim     da
Granja</ows:DeliveryPoint>
  <ows:City>Sao Jose dos Campos</ows:City>
  <ows:AdministrativeArea>Sao Paulo</ows:AdministrativeArea>
  <ows:PostalCode>12245-970</ows:PostalCode>
  <ows:Country>Brasil</ows:Country>
  <ows:ElectronicMailAddress>
       luigi@dpi.inpe.br
  </ows:ElectronicMailAddress>
  </ows:Address>
  </ows:ContactInfo>
  </ows:ServiceContact>
</ows:ServiceProvider>
```

- Seção *OperationsMetadata*

```
<ows:OperationsMetadata>
<ows:Operation name="GetCapabilities">
<ows:DCP>
<ows:HTTP>
  <ows:Get xlink:href="http://note-luigi/cgi-bin/mapserv.exe?" />
  <ows:Post xlink:href="http://note-luigi/cgi-bin/mapserv.exe?" />
  </ows:HTTP>
```

```
  </ows:DCP>
  </ows:Operation>
<ows:Operation name="DetDescribe">
<ows:DCP>
<ows:HTTP>
  <ows:Get xlink:href="http://note-luigi/cgi-bin/mapserv.exe?" />
  <ows:Post xlink:href="http://note-luigi/cgi-bin/mapserv.exe?" />
  </ows:HTTP>
  </ows:DCP>
  </ows:Operation>
<ows:Operation name="GetSegmentation">
<ows:DCP>
<ows:HTTP>
  <ows:Get xlink:href="http://note-luigi/cgi-bin/mapserv.exe?" />
  <ows:Post xlink:href="http://note-luigi/cgi-bin/mapserv.exe?" />
  </ows:HTTP>
  </ows:DCP>
  </ows:Operation>
  </ows:OperationsMetadata>
```

- Seção *Contents*

```
<Contents>
<segmentationBriefing>
  <name>Segmentacao</name>
  <title>Segmentacao de Imagens por Crescimento de Regioes</title>
   <abstract>Crescimento de Regioes eh uma tecnica de agrupamento de
dados, na qual somente as regioes adjacentes, espacialmente, podem ser
agrupadas.</abstract>
  <class>Crescimento de Regioes</class>
  <parameterSchemaURL>http://note-
luigi/wiss/schema.htm</parameterSchemaURL>
<supportedFormats>
  <sourceImageFormat>application/TerraLib</sourceImageFormat>
  <sourceImageFormat>image/GeoTIFF</sourceImageFormat>
  <segCoverageFormat>application/Shapefile</segCoverageFormat>
  <segCoverageFormat>text/gml</segCoverageFormat>
  </supportedFormats>
<KVPParameterList>
<KVPParameter required="true">
  <Keyword>input_image</Keyword>
  <ValueType>text</ValueType>
  </KVPParameter>
<KVPParameter required="true">
  <Keyword>area_min</Keyword>
  <ValueType>integer</ValueType>
  </KVPParameter>
<KVPParameter required="true">
  <Keyword>euc_treshold</Keyword>
  <ValueType>integer</ValueType>
  </KVPParameter>
<KVPParameter required="false">
  <Keyword>exclusion_image</Keyword>
  <ValueType>text</ValueType>
  </KVPParameter>
<KVPParameter required="false">
```

```
  <Keyword>bspline</Keyword>
  <ValueType>boolean</ValueType>
  </KVPParameter>
<KVPParameter required="false">
  <Keyword>chanfer</Keyword>
  <ValueType>boolean</ValueType>
  </KVPParameter>
<KVPParameter required="false">
  <Keyword>compute_stats</Keyword>
  <ValueType>boolean</ValueType>
  </KVPParameter>
  </KVPParameterList>
  </segmentationBriefing>
</Contents>
</Capabilities>
```

4.3 Operação *GetDescribe* (requerido)

A operação *GetDescribe* permite aos clientes do WISS solicitarem informações detalhadas dos serviços suportados pelo WISS. Essas informações são importantes para a maioria das aplicações porque existem diversos algoritmos de segmentação de imagens digitais com diferentes implementações. Os parâmetros usados para configurar o algoritmo de segmentação podem variar de uma implementação para outra. Por esta razão em muitos casos a aplicação cliente do WISS deve requisitar a operação *GetDescribe* antes de requisitar a operação *GetSegmentation*.

4.3.1 *GetDescribe Request*

Um serviço WISS implementa somente um segmentador, desde modo, não é necessário especificar parâmetros mais detalhados para esta operação. Os parâmetros para a operação *GetDescribe Request* são os mesmos apresentados na Tabela 4.1, por exemplo, *Service*, *Request* e *Version*.

4.3.2 *GetDescribe Response*

A resposta ao *GetDescribe request* deve ser uma descrição do segmentador disponível no serviço. As informações retornadas pela operação *GetDescribe* apresentam o nome do segmentador, uma breve descrição, palavras chaves, *links* para documentos com a teoria dos algoritmos, formatos suportados, parâmetros de configuração padrão para o

segmentador. A operação *GetDescribe Response* deve conter os parâmetros listados na Tabela 4.13.

Tabela 4.13 – Subitem *Segmentationbrief*

Nome	Definição	Tipo do Dado	Uso e Multiplicidade
Name	Identificador da segmentador.	String	1 obrigatório
Title	Título para o segmentador.	String	0...1 opcional
Abstract	Descrição do processo de segmentação.	String	0...N opcional
Class	Identificador para a classe do segmentador	String	0...N opcional
ParameterSchemaURL	Endereço do documento que descreve os parâmetros de entrada para o segmentador.	String	1 obrigatório
SourceImageFormat	Formato de entrada das imagens suportadas pela segmentação.	String	1...N obrigatório
SegCoverageFormat	Formato do resultado de retorno da segmentação.	String	1...N obrigatório
KVPParameter	Definição dos parâmetros, quando usado o método HTTP GET.	ver tabela 4.11	1 Obrigatório
SpecificDocumentURL	Endereço do documento com a descrição teórica do algoritmo de segmentação implementado.	String	1 Obrigatório

4.3.3 Exceção

Para a operação *GetDescribe* não é necessário a implementação de exceções, porque não existe a passagem de parâmetros obrigatório para a operação.

4.3.4 Exemplos

4.3.4.1 *GetDescribe Request*

Serviços WISS implementados com transferência HTTP GET, usando passagem de parâmetros pelo URL para a operação *GetDescribe request* usam os parâmetros *Server*,

Request e *Version*. Não existe um parâmetro específico para identificar o segmentador uma vez que cada serviço WISS implementa somente um segmentador.

```
   http://hostname/path?service=WISS&request=GetDescribe&version=0.1
```

Serviços WISS implementados com transferência HTTP POST, usando código XML para a operação *GetDescribe request*.

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDescribe xmlns="http://www.opengis.net/wics"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wics ../wics.xsd"
service="WISS" version="0.1" />
```

4.3.4.2 *GetDescribe Response*

A resposta da operação *GetDescribe* é apresentada abaixo.

```
<?xml version="1.0" encoding="UTF-8" ?>
<SegmentationDescription version="0.1"
xmlns="http://www.opengis.net/wiss"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <name>Segmentacao</name>
  <title>Segmentacao de Imagens por Crescimento de Regioes</title>
  <abstract>Crescimento de Regioes eh uma tecnica de agrupamento de
dados, na qual somente as regioes adjacentes, espacialmente, podem ser
agrupadas.</abstract>
  <class>Crescimento de Regioes</class>
  <parameterSchemaURL>
      http://note-luigi/wiss/schema.htm
  </parameterSchemaURL>
<supportedFormats>
  <sourceImageFormat>application/TerraLib</sourceImageFormat>
  <sourceImageFormat>image/GeoTIFF</sourceImageFormat>
  <segCoverageFormat>application/Shapefile</segCoverageFormat>
  <segCoverageFormat>text/gml</segCoverageFormat>
</supportedFormats>
```

O restante do arquivo XML de resposta descreve os parâmetros necessários para execução do processo de segmentação e o URL para o embasamento teórico do algoritmo implementado.

```
<KVPParameterList>
<KVPParameter required="true">
  <Keyword>input_image</Keyword>
  <ValueType>text</ValueType>
  </KVPParameter>
```

```
<KVPParameter required="true">
  <Keyword>area_min</Keyword>
  <ValueType>integer</ValueType>
  </KVPParameter>
<KVPParameter required="true">
  <Keyword>euc_treshold</Keyword>
  <ValueType>integer</ValueType>
  </KVPParameter>
<KVPParameter required="false">
  <Keyword>exclusion_image</Keyword>
  <ValueType>text</ValueType>
  </KVPParameter>
<KVPParameter required="false">
  <Keyword>bspline</Keyword>
  <ValueType>boolean</ValueType>
  </KVPParameter>
<KVPParameter required="false">
  <Keyword>chanfer</Keyword>
  <ValueType>boolean</ValueType>
  </KVPParameter>
<KVPParameter required="false">
  <Keyword>compute_stats</Keyword>
  <ValueType>boolean</ValueType>
  </KVPParameter>
  </KVPParameterList>
  <SpecificDocumentURL>http://note-
luigi/wiss/espec.htm</SpecificDocumentURL>
</SegmentationDescription>
```

4.4 Operação *GetSegmentation* (requerido)

A operação *GetSegmentation* permite aos clientes do WISS solicitarem a execução do processo de segmentação de imagens digitais. Uma imagem digital pode ser considerada uma matriz bidimensional na qual os índices das linhas e colunas identificam um ponto na imagem e o correspondente valor do elemento da matriz identifica o nível de cinza ou cores no referido ponto. Os elementos desta matriz digital são denominados de *picture element - pixel*. Estas imagens deverão estar disponíveis ou carregadas no mesmo servidor do serviço, para que este possa acessá-las.

4.4.1 *GetSegmentation Request*

Os parâmetros para a operação *GetSegmentation Request* são apresentados na Tabela 4.14 e 4.15. Essa tabela também especifica os tipos e valores, multiplicidade e obrigatoriedade de cada parâmetro listado e uma explicação para todos os parâmetros opcionais que não são incluídos na solicitação da operação.

Tabela 4.14 – Parametros *Getsegmentation Request*

Nome	Definição	Tipo	Valor	Uso e Multiplicidade
Service	Identificado do tipo de serviço.	String	WISS	1 obrigatório
Request	Nome da operação.	String	*GetSegmentation*	1 obrigatório
Version	Versão da operação.	String	a.b	1 obrigatório
SegParameters	Parâmetros necessários para execução do serviço.	String	ver tabela 4.15	1...N opcional
FormatInput	Formato da imagem de entrada.	String	Escolher um dos formatos listado no *SourceImageFormat* do *GetCapabilities*	1 obrigatório
FormatExclude	Formato da imagem de exclusão.	String	Escolher um dos formatos listado no *SourceImageFormat* do *GetCapabilities*	1 opcional
SegCoverageFormat	Formato do resultado da segmentação.	String	Escolher um dos formatos listado no *SegCoverageFormat* do *GetCapabilities*	1 opcional, o formato padrão está descrito na operação *GetDescribe*

Tabela 4.15 – Subitem *Segparameters*

Nome	Definição	Tipo	Valor	Uso e Multiplicidade
Keyword	Nome do parâmetro.	String	O nome dos parametros estão listados na operação *GetDescribe*.	1 obrigatório
Value	Valor correspondente ao parâmetro.	String	O tipo do valor dos parametros estão listados na operação *GetDescribe*.	1 obrigatório

4.4.2 *GetSegmentation Response*

A resposta normal para um operação *GetSegmentation* é um URL apontando para a imagem segmentada e o formato dela. Segue abaixo o esquema XML especificando o conteúdo e estrutura de uma resposta da operação *GetSegmentation*.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Segmentation version="0.1" xmlns="http://www.opengis.net/wiss"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<segmentedCoverage xlink:href="http://note-luigi/wiss/data/tmp/">
  <ows:Name>wissdata-20060218-215502</ows:Name>
  <ows:Format>application/Shapefile</ows:Format>
  </segmentedCoverage>
</Segmentation>
```

4.4.3 Exceção

Quando o servidor de *Web* encontra um erro na execução na operação *GetSegmentation* ele deve retornar uma exceção reportando uma mensagem específica do erro. A Tabela 4.16 apresenta o código de cada exceção, a provável causa e a explicação (mensagem de retorno).

Tabela 4.16 – Exceção *Getsegmentation*

Valor	Mensagem	Causa
MissingSourceImage	*Operation request does not include a parameter value*	Imagem não encontrada.
InvalidSourceImageURI	*Operation request contains an invalid parameter value*	Parâmetro inválido.
InvalidFormat	*The format name for the input source image is not supported by the server.*	Formato de entrada inválido.
InvalidSegCoverageFormat	*The format name for the output segmentation coverage is not supported by the server.*	Formato de saída inválido.
InvalidSegParameters	*Specified value for the SegParameters inconsistent with that required by the segmentation.*	Parâmetro inválido.

4.4.4 Exemplos

4.4.4.1 *GetSegmentation Request*

Serviços WISS implementados com transferência HTTP GET, usando passagem de parâmetros pelo URL para a operação *GetSegmentation request* usam os parâmetros *Server*, *Request*, *Version* e os parâmetros listados na operação *GetDescribe*. Não existe um parâmetro específico para identificar o segmentador uma vez que cada serviço WISS implementa somente um segmentador.

```
http://note-luigi/cgi-
bin/mapserv.exe?service=wiss&version=0.1&request=getsegmentation&input
_image=cbers_b2_crop_C.tif&area_min=50&euc_treshold=10
```

Serviços WISS implementados com transferência HTTP POST, usando código XML para a operação *GetSegmentation request*.

```
<?xml version="1.0" encoding="UTF-8"?>
<GetSegmentation xmlns="http://www.opengis.net/wics"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wiss ../wics.xsd"
service="WISS" version="0.1">
      <FormatInput> image/GeoTIFF </FormatInput>
      <FormatInput> application/TerraLib </FormatInput>
      <SegCoverageFormat> application/Shapefile </SegCoverageFormat>
      <SegParameters>
            <Parameter>
                  <Keyword> input_image </Keyword>
                  <Value> cbers_b2_crop_C.tif </Value>
            </Parameter>
            <Parameter>
                  <Keyword> area_min </Keyword>
                  <Value> 50 </Value>
            </Parameter>
            <Parameter>
                  <Keyword> euc_treshold </Keyword>
                  <Value> 10 </Value>
            </Parameter>
      </SegParameters>
</GetSegmentation>
```

4.4.4.2 *GetSegmentation Response*

A resposta para uma solicitação de segmentação de imagens, se não ocorrer nenhuma exceção é a seguinte:

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

```
<Segmentation version="0.1" xmlns="http://www.opengis.net/wiss"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<segmentedCoverage xlink:href="http://note-luigi/wiss/data/tmp/">
  <ows:Name>wissdata-20060218-215502</ows:Name>
  <ows:Format>application/Shapefile</ows:Format>
  </segmentedCoverage>
<segmentedCoverage xlink:href="http://note-luigi/cgi-
bin/mapserv.exe?">
<KVPParameterList>
<KVPParameter required="true">
  <Keyword>map</Keyword>
<Value>C:\ApacheGroup\Tomcat5027\webapps\wiss\config\config.map</Value
>
  </KVPParameter>
<KVPParameter required="true">
  <Keyword>service</Keyword>
  <Value>WISS</Value>
  </KVPParameter>
<KVPParameter required="true">
  <Keyword>version</Keyword>
  <Value>0.1</Value>
  </KVPParameter>
<KVPParameter required="true">
  <Keyword>request</Keyword>
  <Value>GetFeature</Value>
  </KVPParameter>
<KVPParameter required="true">
  <Keyword>typename</Keyword>
  <Value>wissdata-20060218-215502</Value>
  </KVPParameter>
  </KVPParameterList>
  </segmentedCoverage>
</Segmentation>
```

4.5 Operação *GetFeature* (opcional)

A operação *GetFeature* retorna instâncias dos objetos resultantes do processo de segmentação disponíveis na base de dados em formato GML. O cliente pode selecionar quais objetos deseja por critérios espaciais ou não.

A operação *GetFeature* faz parte da implementação da especificação *OpenGIS Web Feature Service* (WFS) (Vretanos, 2005) que define um serviço para que clientes possam recuperar objetos (*features*) espaciais em formato GML de servidores WFS.

Esta operação poderá ser utilizada quando um WFS estiver implementado no servidor, caso contrário os objetos resultantes da segmentação deverão ser recuperados via

download pelo URL descrito no documento XML retornado pela operação *GetSegmentation* da especificação WISS.

4.6 WISS e OGC *Web Service*

Neste capítulo foram apresentados as 4 operações implementadas no serviço que constitui a especificação completa para o WISS seguindo as normas de interface de serviços do OWS. As operações permitem aos clientes solicitar e receber de volta os metadados do serviço; permite solicitar informações detalhadas dos segmentadores disponíveis no serviço; permite solicitar a execução do serviço de segmentação enviando os parâmetros necessários para o segmentador, em resposta o serviço retorna um arquivo XML com o URL do arquivo de saída nos formatos GML ou *Shapefile*; permite ao cliente solicitar e receber de volta a imagem rotulada no formato GML.

O conteúdo e organização da operação *getCapabilities* é o mesmo para todos os OWS especificados pelo OGC, com exceção da seção *Contents* que define alguns parâmetros específicos e obrigatórios para o serviço WISS. A operação *getDescribe* é específica para o serviço WISS seu conteúdo e organização são bem parecidos com a seção *Contents* especificado na operação *getCapabilities*, existem mais alguns parâmetros específicos para a operação *getDescribe* que definem o URL do documento que descreve os parâmetros de entrada do segmentador e o URL do documento com o embasamento teórico do algoritmo de segmentação implementado. A operação *getSegmentation* é específica para o serviço WISS e define os parâmetros necessários para execução do processo de segmentação, em resposta é retornado um arquivo XML com o URL para *download* do arquivo de saída no formato *Shapefile* e os parâmetros necessários para execução da operação *getFeature* que retornará o resultado do processamento no formato GML. A operação *getFeature* faz parte da implementação da especificação WFS que define um serviço para clientes recuperarem objetos (*features*) espaciais em formato GML, esta operação poderá ser utilizada quando o WFS estiver implementado no servidor.

No próximo capítulo é apresentada a implementação de um protótipo de sistema com base na especificação WISS. Nele estão integrados a biblioteca TerraLib onde estão

disponíveis os algoritmos de segmentação e o MapServer utilizado como *front-end* para acesso aos serviços disponibilizados e ao banco de imagens.

CAPÍTULO 5

WISS - *WEB IMAGE SEGMENTATION SERVICE*: IMPLEMENTAÇÃO

5.1 Visão Geral

Para a construção de um protótipo é importante observar algumas considerações relacionadas abaixo:

- Interoperabilidade do Sistema;
- Interoperabilidade do *Web Service*;
- Reuso do software;
- Capacidade de expansão do sistema.

Entende-se por interoperabilidade de um sistema a capacidade que este provê para acessar e processar dados e serviços de diferentes fontes. A implementação de um *Web Service* geográfico interoperável deve ser baseada em padrões e fornecer recursos para acessar dados e serviços remotos (Percivall, 2003).

Um software reutilizável é a forma em que todos os serviços *Web* geográficos e modelos geográficos são projetados para serem dinamicamente reutilizáveis por outros sistemas e serviços.

A capacidade de expansão do sistema depende basicamente da sua arquitetura. Um sistema baseado em componentes, com cada componente utilizando padrões de interface e com arquitetura modular, não terá dificuldade para se expandir.

Com base nisto, este trabalho apresenta aspectos inerentes à utilização das tecnologias *Web Service* como forma de permitir o processamento de imagens de sensoriamento remoto, utilizando-se a Internet como canal de comunicação e as especificações de *Web Services* do Consórcio *OpenGIS* como base para o protótipo do sistema proposto.

As ferramentas escolhidas para implementação do WISS são a biblioteca TerraLib, onde estão disponíveis os algoritmos de segmentação, e o MapServer utilizado como *front-end* de aplicações espaciais na *Web*, responsável por receber solicitações de execução do serviço WISS e apresentar os resultados em forma de mapas.

A arquitetura proposta para o desenvolvimento do protótipo é ilustrada na Figura 5.1, os componentes representados pelas elipses não foram implementados na versão atual do protótipo do sistema desenvolvido, ficando estes, como proposta para novas versões do protótipo onde poderá ser implementado o suporte para o serviço WISS aos padrões do W3C.

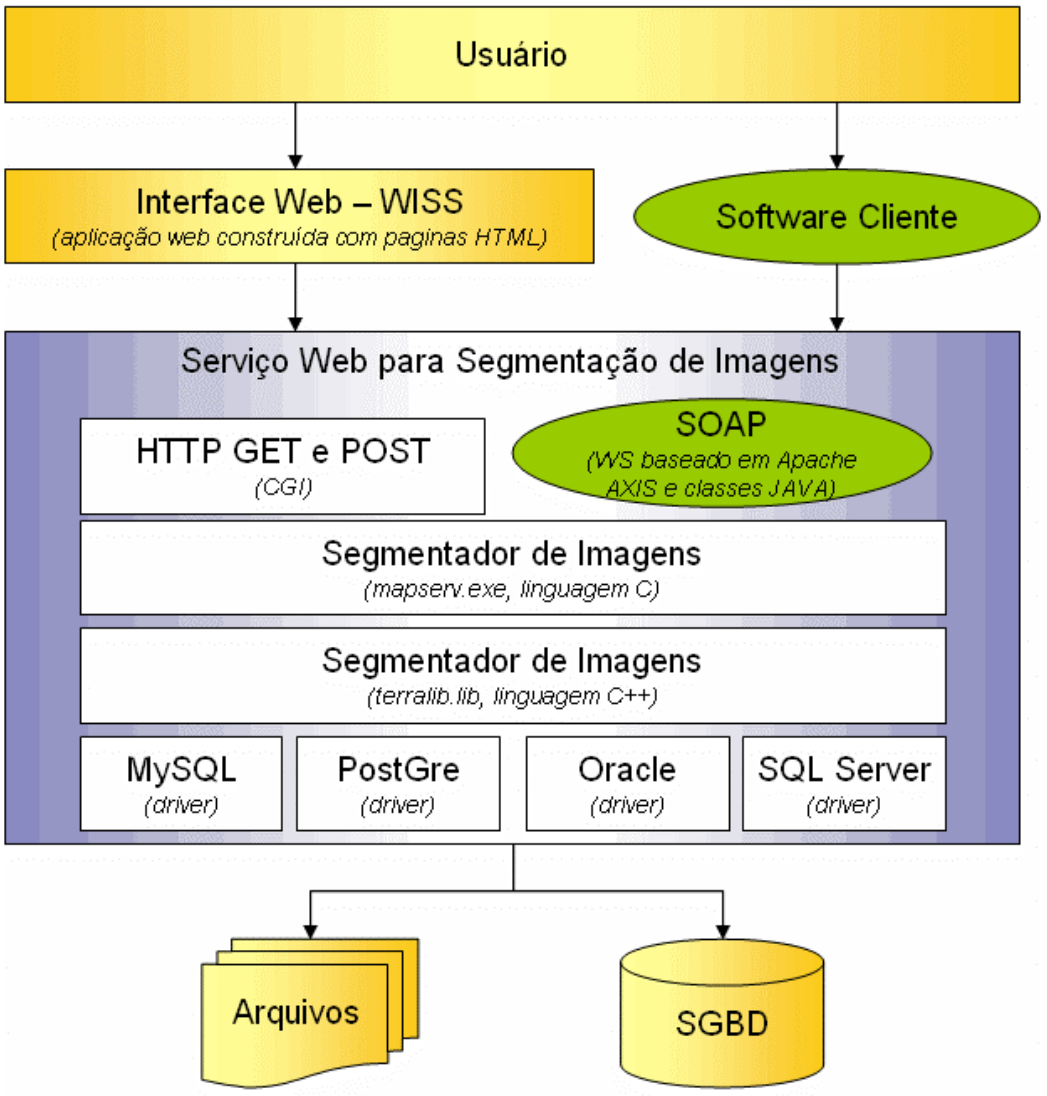

Figura 5.1 – Arquitetura Proposta

O Servidor *Web* recebe dos diversos clientes as requisições sob a forma de um *Uniform Resource Locator* (URL) transportados pelo protocolo HTTP GET/POST ou XML encapsuladas em mensagens SOAP e as encaminha para o serviço responsável pela sua execução. O serviço responsável processa a requisição, acessando informações na base de dados ou arquivos de sistema e envia a resposta de volta para o Servidor *Web*, que a codifica como uma resposta XML e a envia para a aplicação cliente. Esta, por sua vez, decodifica a resposta XML e aplica as funções de apresentação apropriadas para mostrar a resposta nos dispositivos. Em uma arquitetura baseada em serviços, vale ressaltar que um serviço pode acessar outro a fim de executar suas funções. Desta forma é gerado um encadeamento de serviços, podendo um mesmo serviço assumir o papel de provedor ou consumidor.

O protótipo de sistema foi modelado seguindo a *Unified Modeling Language* (UML) que é uma linguagem de modelagem padronizada para engenharia de sistemas orientados a objeto (Rumbaugh et al., 2000). Ela foi concebida a partir de várias técnicas de projeto e análise orientados a objeto existentes, de vários autores, e principalmente, a partir do "processo unificado", de Booch, Rumbaugh e Jacobson (Rumbaugh et al., 1999).

A *Object Management Group* (OMG) (OMG, 2004) inclui nas normas da UML uma sugestão de processo ou metodologia de desenvolvimento de sistemas orientados a objetos, no qual pretende enfocar os pontos onde efetivamente a UML pode ser de grande valia para o desenvolvedor. Este processo está dividido em 4 fases:

- Concepção, nesta dissertação chamada de Visão de Negócio;
- Elaboração, nesta dissertação chamada de Visão da Análise;
- Construção, nesta dissertação chamada de Projeto;
- Transição, nesta dissertação chamada de Visão da Implantação.

5.2 Visão do Negócio

Este tópico apresenta a modelagem dos aspectos relacionados ao protótipo, descreve brevemente as principais funcionalidades envolvidas, os atores que participam do negócio, a relação entre os atores e funcionalidades e a ordem em que às ações são executadas no desenrolar do negócio. A Figura 5.2 representa o Diagrama de Caso de Uso (Rumbaugh et al., 2000).

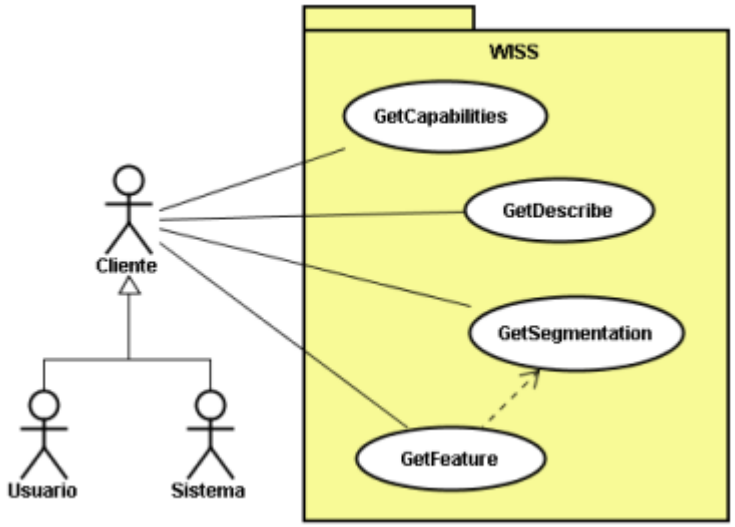

Figura 5.2 - Diagrama de Caso de Uso

Os atores representados pelo esquema de um boneco, representam um elemento do mundo real que executa alguma função no sistema. Este elemento pode ser uma pessoa, um sistema, uma entidade externa como um dispositivo ou outro sistema computacional.

Os casos de uso representados por elipses, representam um serviço ou uma funcionalidade que o sistema presta visivelmente percebidas pelo ambiente externo, que pode ser considerada uma ação que um ator executa.

Caso de Uso:	***GetCapabilities***
Descrição:	Implementação obrigatória, esta operação permite ao cliente solicitar e receber de volta o metadados do serviço com informações da organização detentora do serviço, sobre as operações especificadas e implementadas no servidor e o resumo dos segmentadores disponíveis.
Atores:	Cliente
Pré-Condição:	
Fluxo Principal:	1. O ator envia requisições sob a forma de um URL solicitando a execução da operação *GetCapabilities*. 2. O sistema processa a requisição e envia a resposta de volta para o Servidor *Web*, que a codifica como uma resposta XML e a envia para a aplicação cliente. 3. O ator recebe o arquivo XML descrevendo os metadados do serviço.
Fluxo Alternativo e Exceções:	2a. O sistema processa a requisição, mas não encontra parâmetros necessários para execução da operação. 2b. O ator recebe arquivo XML descrevendo o erro ocorrido na execução do serviço.
Pós-Condição:	Arquivo XML enviado para a aplicação cliente.

Caso de Uso:	***GetDescribe***
Descrição:	Implementação obrigatória, esta operação permite ao cliente solicitar informações detalhadas do segmentador disponível, o servidor responde a esta solicitação com um documento em XML descrevendo em detalhes o segmentador implementado no serviço. Essas informações incluem uma descrição teórica do algoritmo do segmentador, tipos de dados suportados, os parâmetros específicos solicitados por cada segmentador e os valores padrão para esses parâmetros em caso do cliente não enviar os valores.
Atores:	Cliente
Pré-Condição:	
Fluxo Principal:	1. O ator envia requisições sob a forma de um URL solicitando a execução da operação *GetDescribe*. 2. O sistema processa a requisição e envia a resposta de volta para o Servidor *Web*, que a codifica como uma resposta XML e a envia para a aplicação cliente. 3. O ator recebe o arquivo XML, com informações detalhadas do segmentador.
Fluxo Alternativo e Exceções:	2a. O sistema processa a requisição, mas não encontra parâmetros necessários para execução da operação. 2b. O ator recebe arquivo XML descrevendo o erro ocorrido na

	execução do serviço.
Pós-Condição:	Arquivo XML enviado para a aplicação cliente.

Caso de Uso:	***GetSegmentation***
Descrição:	Implementação obrigatória, esta operação permite ao cliente executar o serviço de segmentação, enviando os parâmetros requeridos pelo serviço. O servidor responde a solicitação retornando um XML com o URL do arquivo de saída no formato *Shapefile*. O acesso ao resultado do processamento é feito através de *download* do arquivo de saída.
Atores:	Cliente
Pré-Condição:	
Fluxo Principal:	1. O ator envia requisição sob a forma de um URL solicitando a execução da operação *GetSegmentation*. 2. O sistema processa a requisição, acessando informações na base de dados ou arquivos de sistema, executa o processo de segmentação de imagens e envia a resposta de volta para o Servidor *Web*, que a codifica como uma resposta XML e a envia para a aplicação cliente. 3. O ator recebe o arquivo XML com o URL do arquivo de saída.
Fluxo Alternativo e Exceções:	2a. O sistema processa a requisição, mas não encontra parâmetros necessários para execução da operação ou os parâmetros informados são inválidos. 2b. O ator recebe arquivo XML descrevendo o erro ocorrido na execução do serviço.
Pós-Condição:	Arquivo XML enviado para a aplicação cliente.

Caso de Uso:	***GetFeature***
Descrição:	Implementação opcional, esta operação permite ao cliente solicitar e receber de volta a feição rotulada no formato GML.
Atores:	Cliente
Pré-Condição:	Serviço WFS implementado no servidor.
Fluxo Principal:	1. O ator envia requisições sob a forma de um URL solicitando a execução da operação *GetSegmentation*. 2. O sistema processa a requisição, recupera a imagem rotulada armazenada no formato *shapefile*, executa o processo de exportação e envia a resposta de volta para o Servidor *Web*, que a codifica como uma resposta GML e a envia para a aplicação cliente. 3. O ator recebe a imagem rotulada formatada em GML.
Fluxo	2a. O sistema processa a requisição, mas não encontra parâmetros

Alternativo e Exceções:	necessários para execução da operação ou os parâmetros informados são inválidos. 2b. O ator recebe arquivo XML descrevendo o erro ocorrido na execução do serviço.
Pós-Condição:	Arquivo XML enviado para a aplicação cliente.

5.3 Visão da Análise

Este tópico apresenta a forma que os casos de uso podem ser realizados, o diagrama de seqüência representa as classes que colaboram para a realização de um caso de uso em uma ordem seqüencial e temporal. É uma representação da interação entre objetos (comportamento) ao longo do tempo, formando uma seqüência de eventos através de chamadas de serviço (métodos).

Cada objeto da interação é colocado na horizontal de forma que as mensagens possam fluir de uma para a outra descrevendo o fluxo de eventos. Um ator é representado para apresentar a interação com o sistema, principalmente para iniciar o fluxo de eventos.

Elementos estereotipados como fronteiras, controle e entidade são utilizados para representar telas / relatórios, processamento e itens de banco de dados. Cada elemento possui uma linha vertical chamada de linha do tempo, mostrando seu tempo de vida na interação.

As Figuras 5.3 a 5.6 ilustram a seqüência de execução das 4 operações especificadas no serviço WISS.

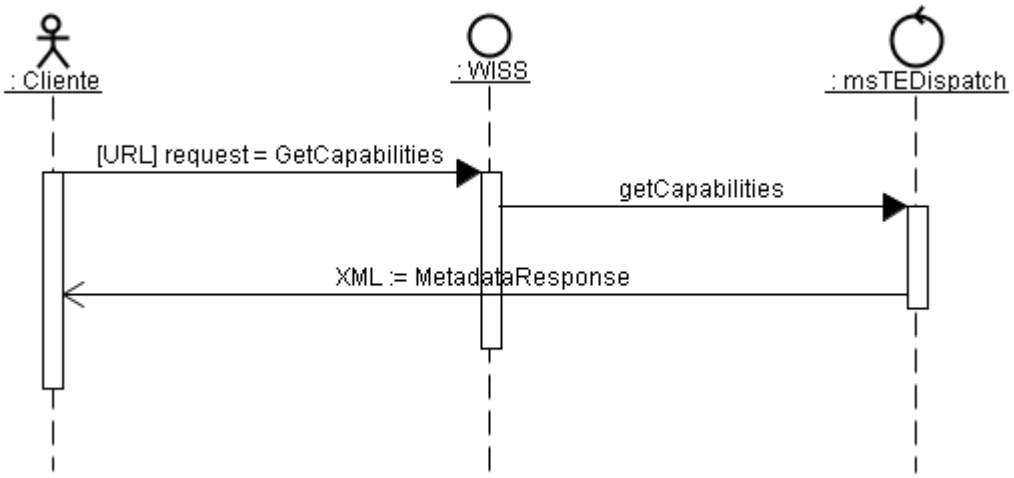

Figura 5.3 - Diagrama de Sequência, *getCapabilities*

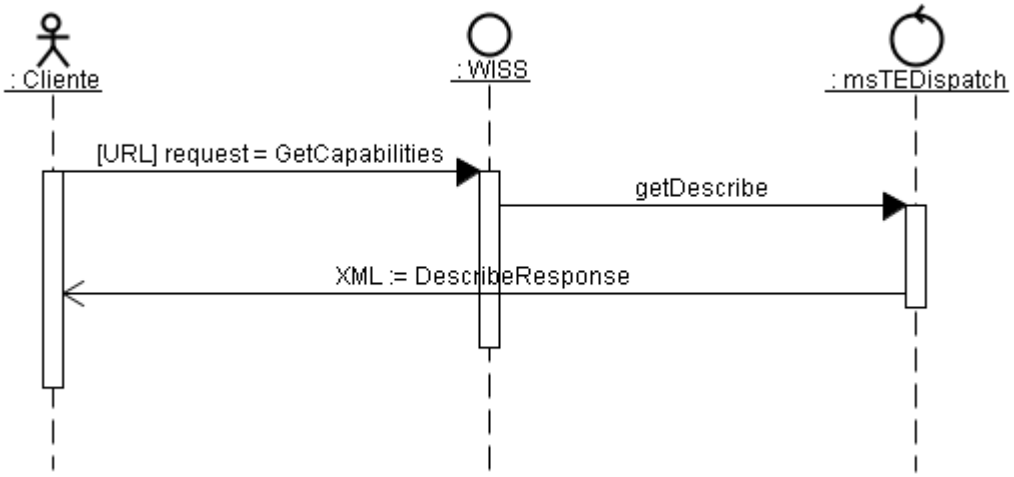

Figura 5.4 - Diagrama de Sequência, *getDescribe*

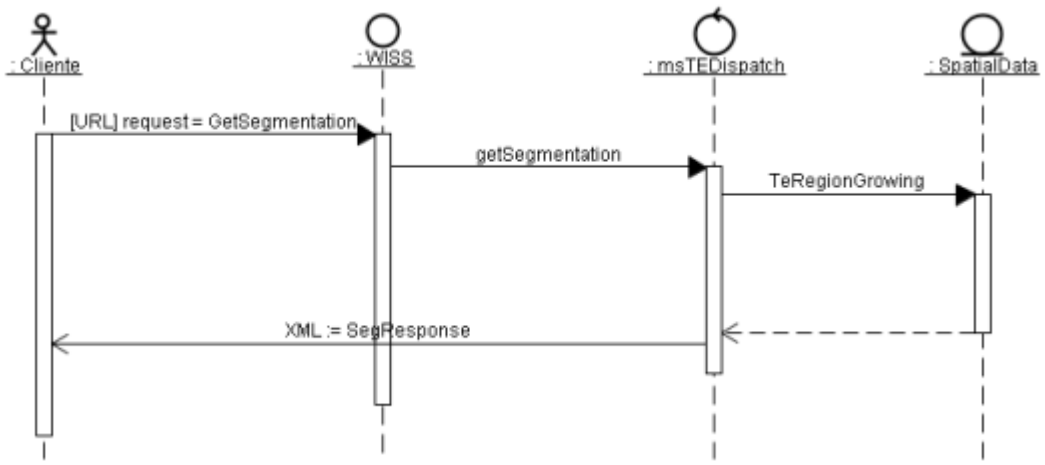

Figura 5.5 - Diagrama de Sequência, *getSegmentation*

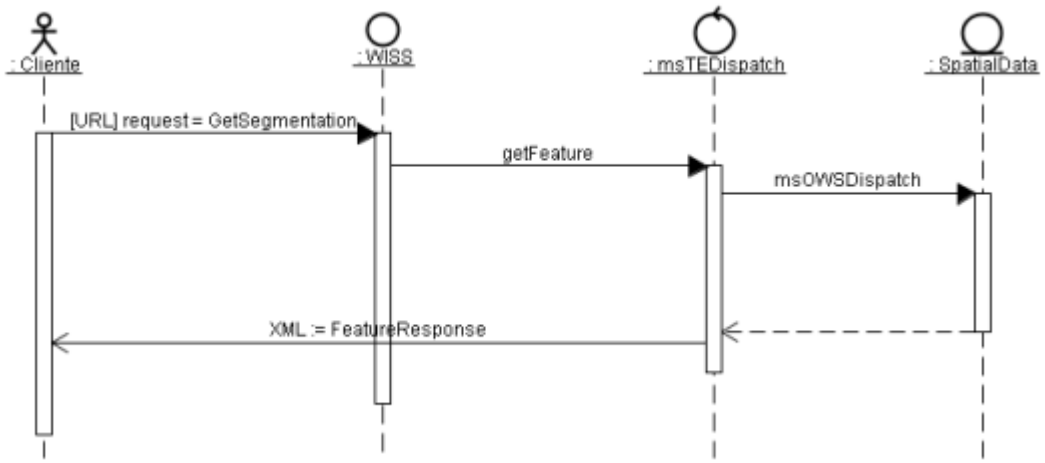

Figura 5.6 - Diagrama de Sequência, *getFeature*

5.4 Projeto: Classes

As classes estereotipadas aparecem na fase de análise (visão do negócio e análise), nos diagramas de caso de uso e seqüência. Podemos definir classe como sendo uma especificação onde estão encapsulados as informações e os comportamentos dos objetos, e objeto é uma instância em tempo de execução de uma classe. As Figuras 5.7 a 5.9 apresentam o diagrama de classes do WISS, o serviço é uma composição de um conjunto de operações e tipos de dados.

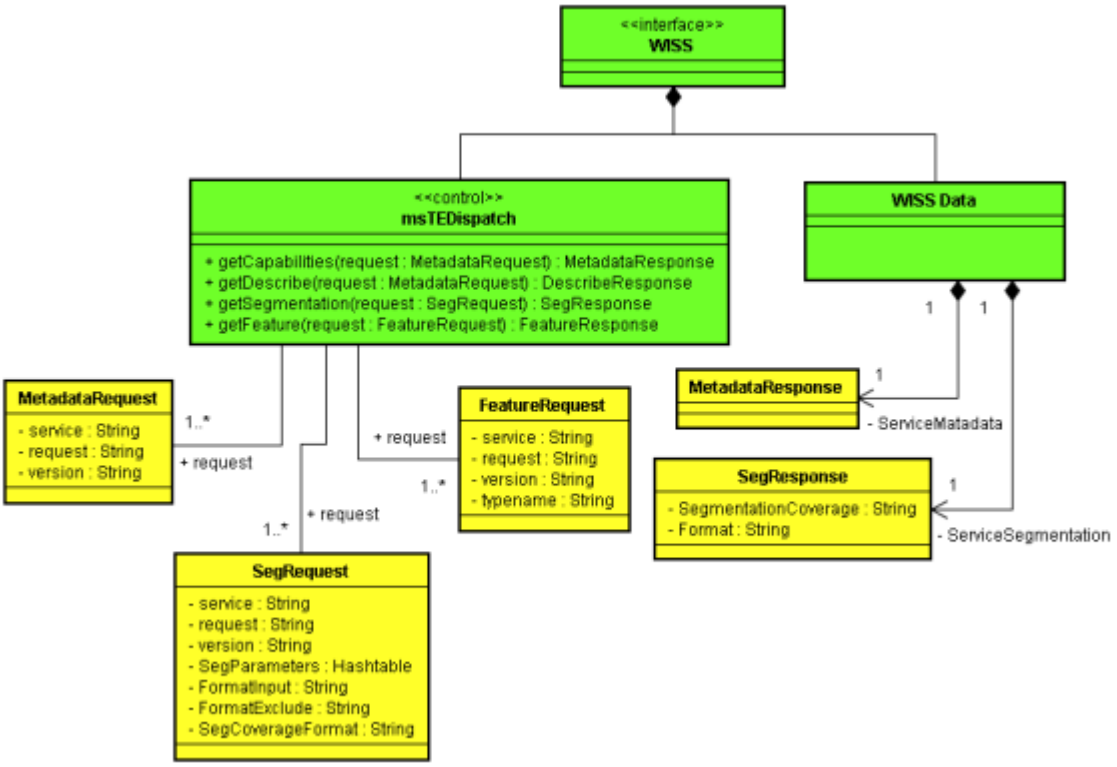

Figura 5.7 - WISS Serviço *Web* para Segmentação de Imagens

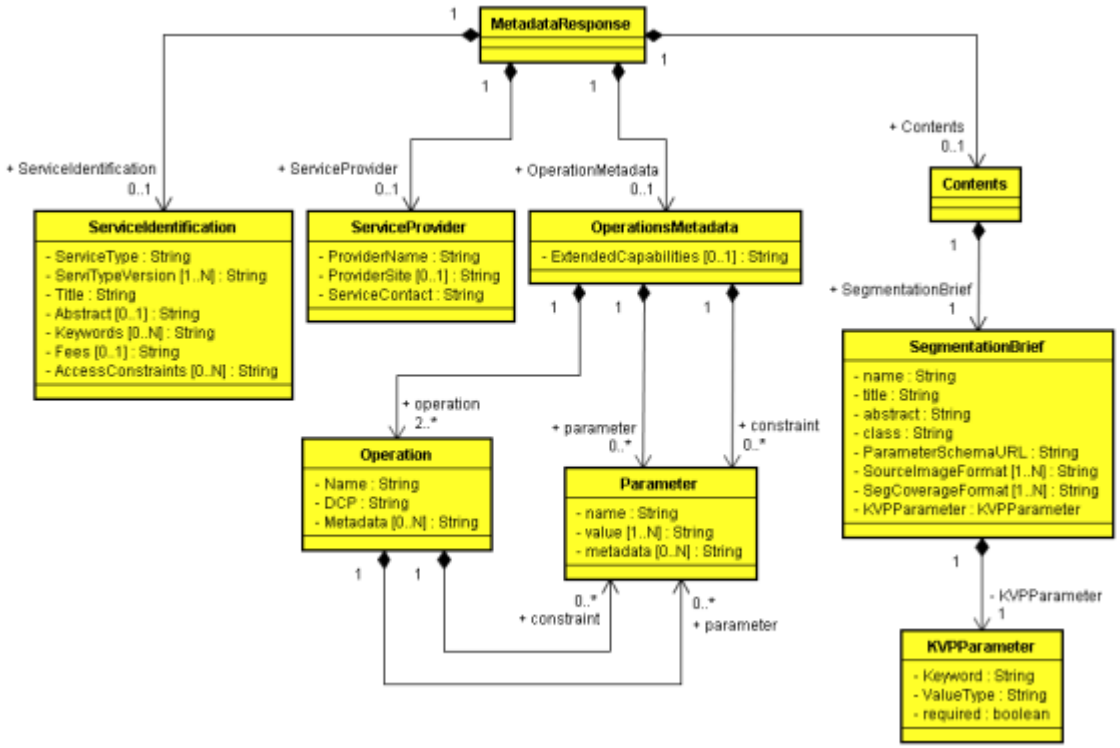

Figura 5.8 - WISS *GetCapabilities Response*

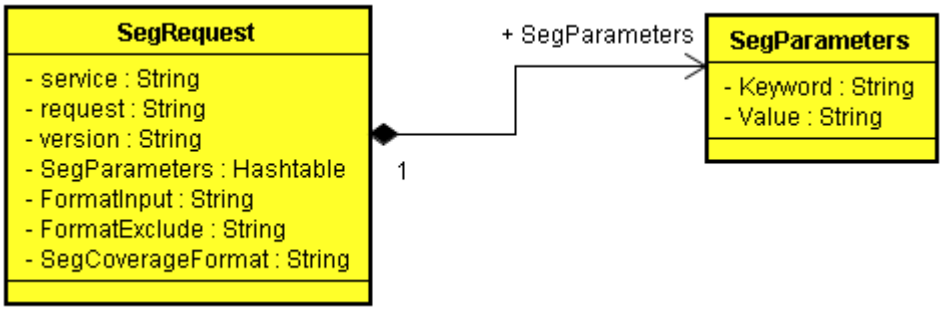

Figura 5.9 - WISS *GetSegmentation Request*

5.5 Projeto: Componentes

Este tópico apresenta a modelagem dos aspectos físicos do protótipo onde serão representados pelo diagrama de componentes. Componentes são materializações de elementos lógicos com classes e interfaces, são utilizados para residirem em um nó (parte do sistema) como: executáveis, bibliotecas, tabelas, arquivos e documentos. Um

componente é uma parte física e substituível de um sistema ao qual se adapta e fornece a realização de um conjunto de interfaces. A Figura 5.10 apresenta os componentes utilizados na implementação do protótipo.

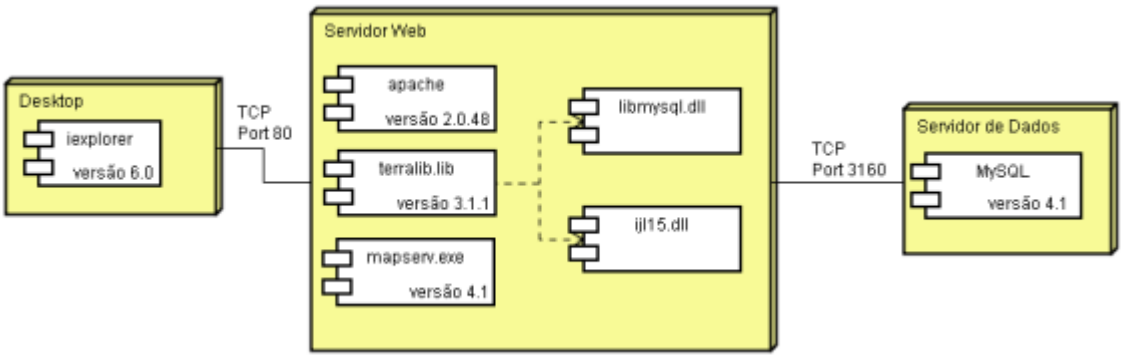

Figura 5.10 - Diagrama de Componentes

5.6 Visão da Implantação: Protótipo de Sistema

Este tópico apresenta o protótipo de sistema implementado seguindo a especificação e arquitetura apresentadas ao longo desse trabalho. A Figura 5.10 ilustra os componentes utilizados no protótipo, o banco de imagens foi criado e gerenciado com o auxílio do aplicativo TerraView onde foram importadas imagens CBERS para o MySQL 4.1.

A Figura 5.11 apresenta a interface principal do protótipo, onde está dividida em três partes:

1) A área identificada com o número 1 (um) permite ao usuário fornecer os parâmetros necessários para a requisição de um serviço. O parâmetro *Request* identifica a operação que será executada;

2) A área identificada com o número 2 (dois) permite ao usuário enviar uma imagem no formato *GeoTiff* para o servidor clicando no botão *"Carregar..."* ou selecionar uma que já esteja disponível no servidor clicando no botão "*Catalogo...*". Esta funcionalidade pode ser utilizada tanto para carregar/selecionar imagens de entrada como de exclusão;

3) A área identificada com o número 3 (três) apresenta o resultado do processamento da operação solicitada. Geralmente o retorno das operações são arquivos XML onde estão armazenados os resultados.

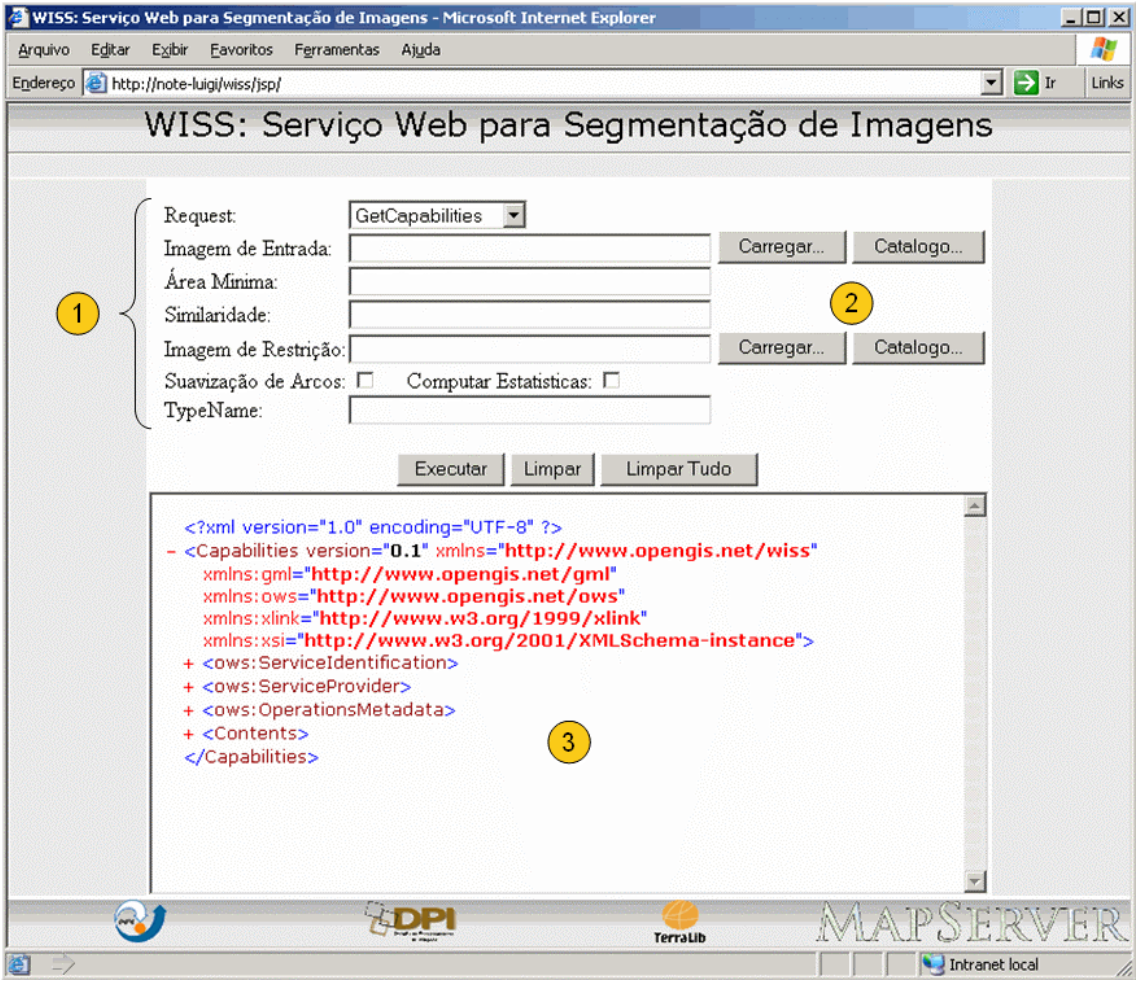

Figura 5.11 – Interface Principal do Protótipo

A Figura 5.12 apresenta as opções disponíveis no protótipo para o usuário fornecer ou selecionar as imagens que serão utilizadas no processo de segmentação. As opções disponíveis são:

- Botão Carregar: permite ao usuário enviar uma imagem ao servidor para o processamento da mesma.

- Botão Catálogo: permite ao usuário selecionar uma imagem disponível no repositório do servidor.

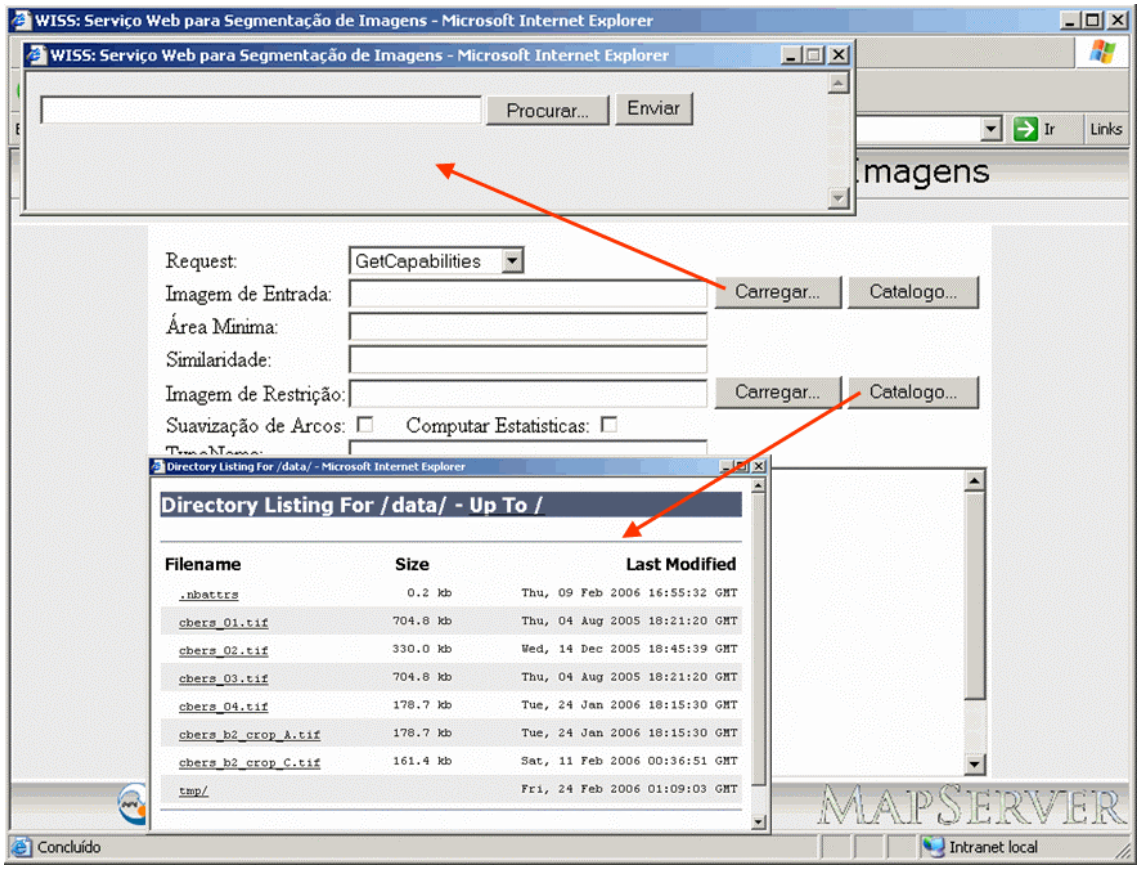

Figura 5.12 – Selecionar imagens de entrada ou exclusão

A Figura 5.13 apresenta o arquivo XML retornado pelo serviço WISS após execução da operação *getSegmentation*, na *tag* <*segmentedCoverage*> é informado o endereço URL onde esta disponível o arquivo criado após a execução da operação ,na *tag* <*ows:Name*>

é informado o nome do arquivo de saída que é composto pela data e hora que a operação foi solicitado, na *tag <ows:Format>* é informado o formato do arquivo.

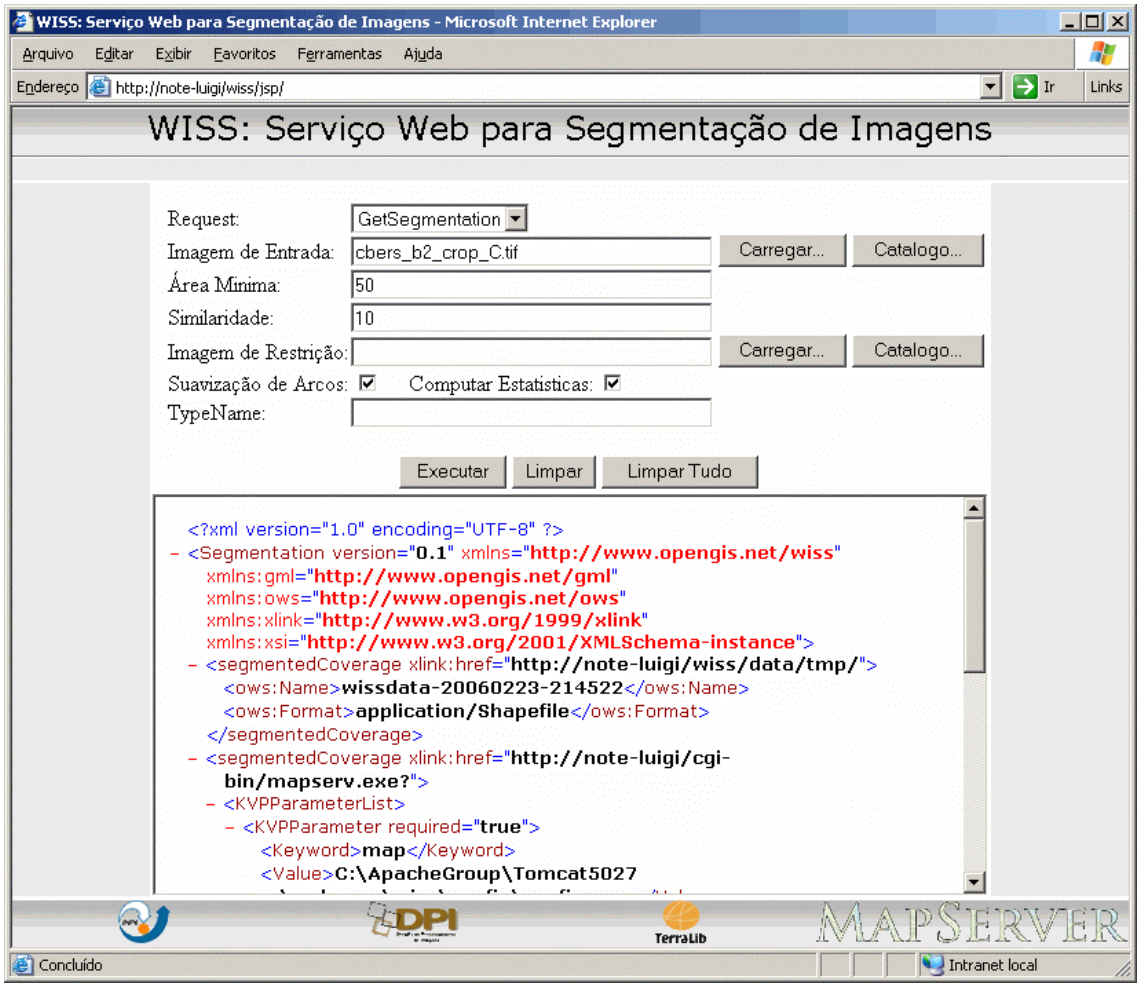

Figura 5.13 – Resultado da operação *getSegmentation*

A Figura 5.14 apresenta o arquivo GML retornado pelo serviço WISS após execução da operação *getFeature*, para cada região ou feição geográfica exportada para o arquivo GML são fornecidas as informações alfanuméricas como código, media e variância, a projeção e as coordenadas geográficas de cada vértices que formam a região.

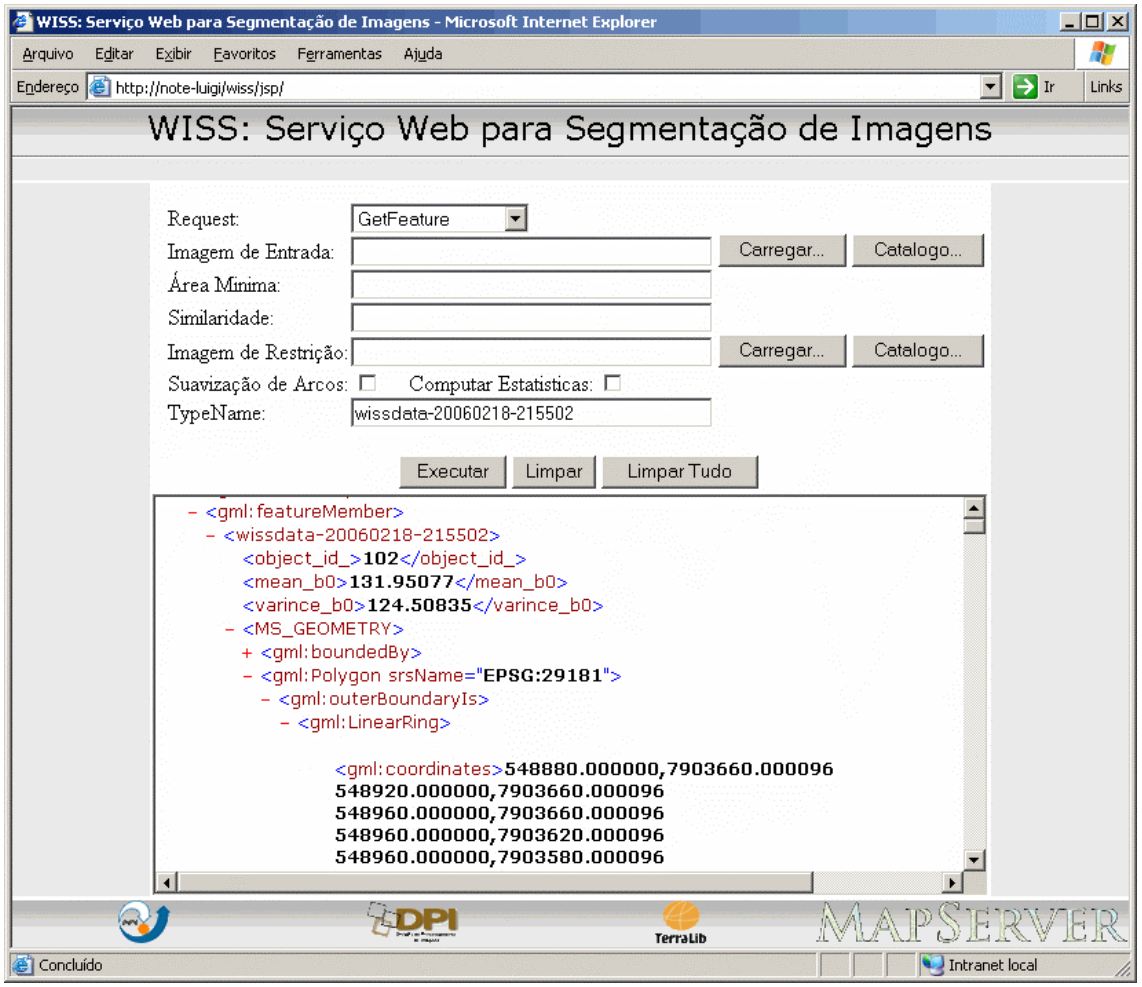

Figura 5.14 – Resultado da operação *getFeature* em GML

A Figura 5.15 apresenta o resultado retornado pelo serviço WMS após execução da operação *getMap*, este operação não faz parte da especificação do serviço WISS, mas como ambos os serviços seguem as especificações propostas pelo OGC o resultado de um serviço pode ser utilizado para execução de um outro serviço.

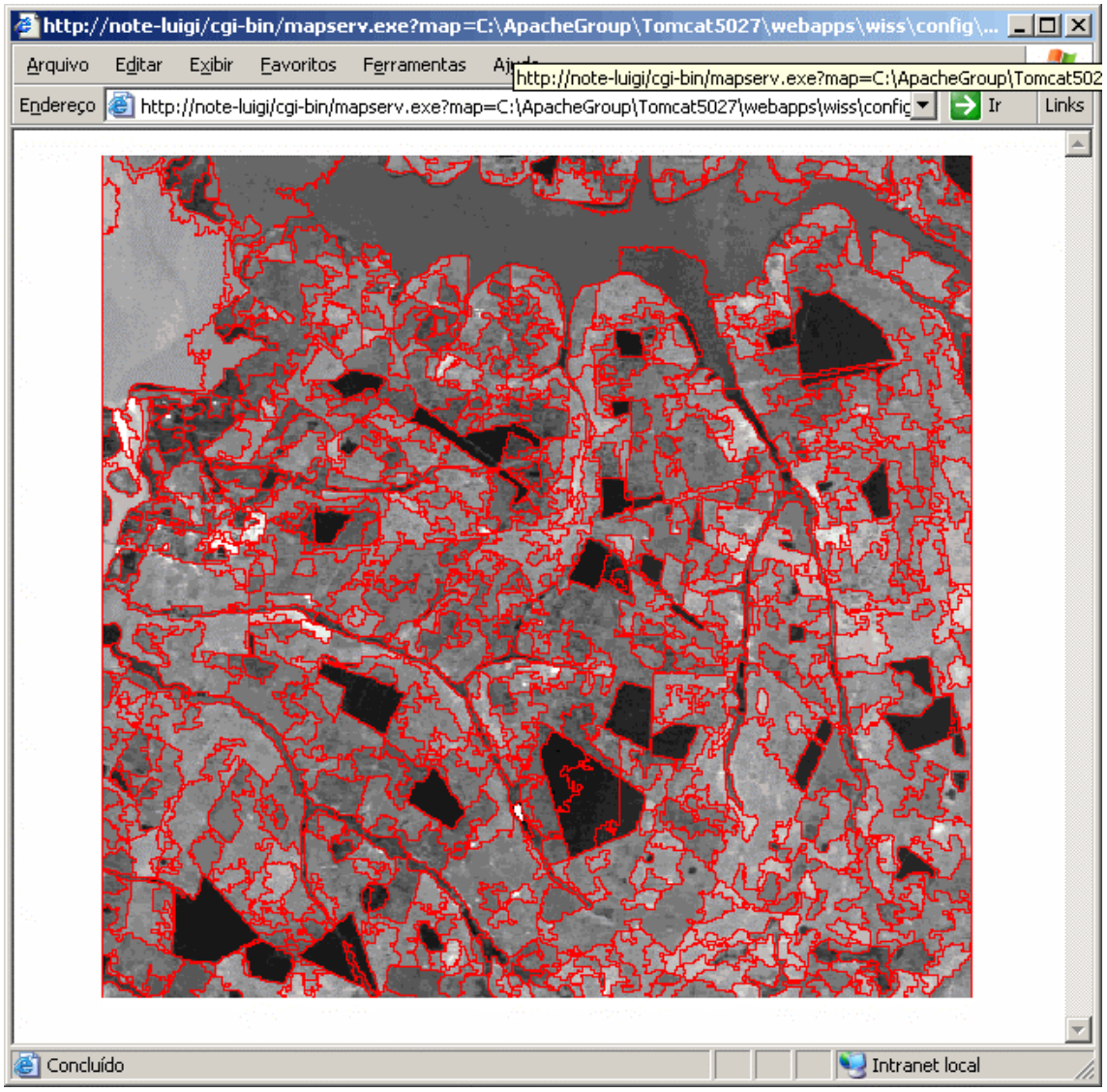

Figura 5.15 – Resultado da operação getMap do serviço WMS

A Figura 5.16 apresenta o acesso ao endereço URL fornecido pelo arquivo XML retornado após execução da operação *getSegmentation*, a imagem rotulada criada pelo processo de segmentação esta disponível para *download*.

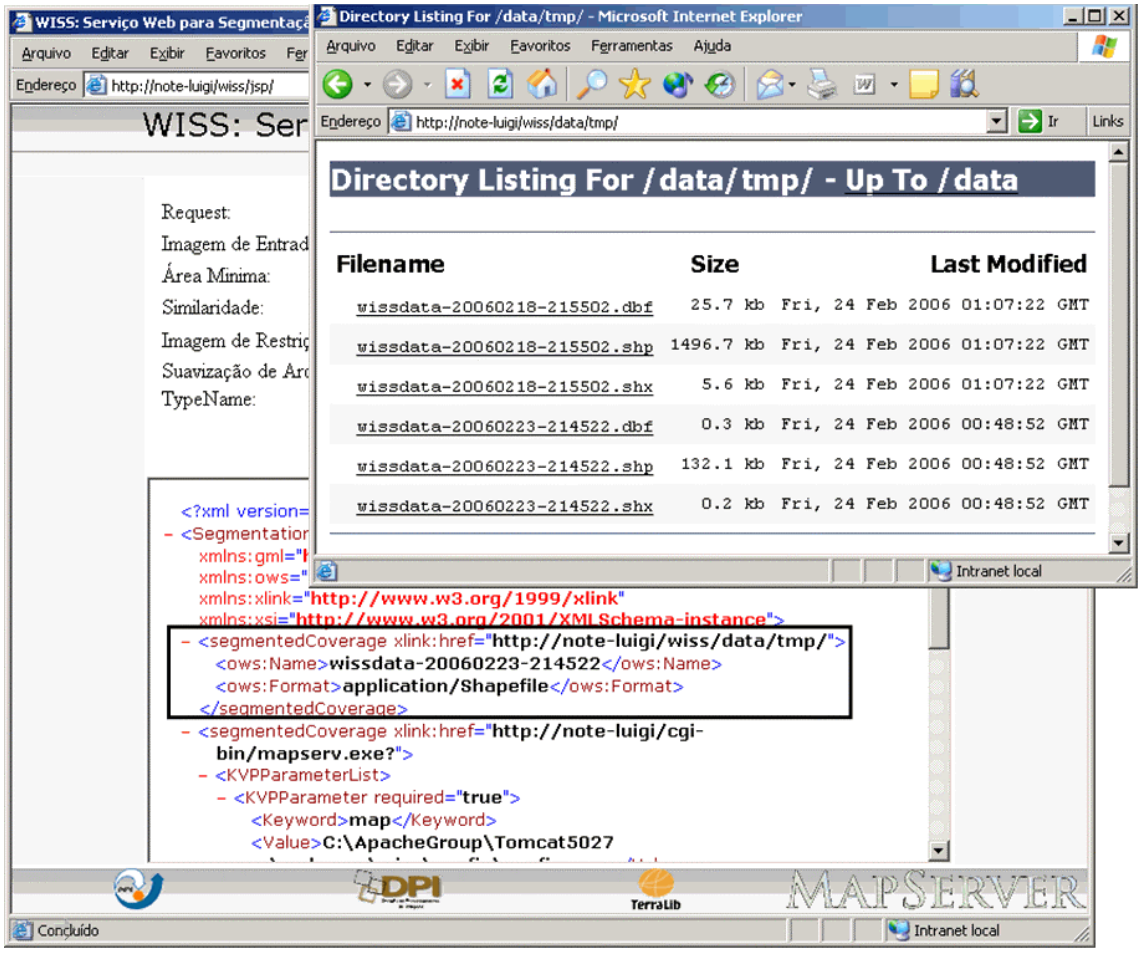

Figura 5.16 – Resultado disponível para *download* da operação *getSegmentation*

A Figura 5.17 apresenta o resultado do processo de segmentação do serviço WISS sendo acessado pela aplicação TerraView.

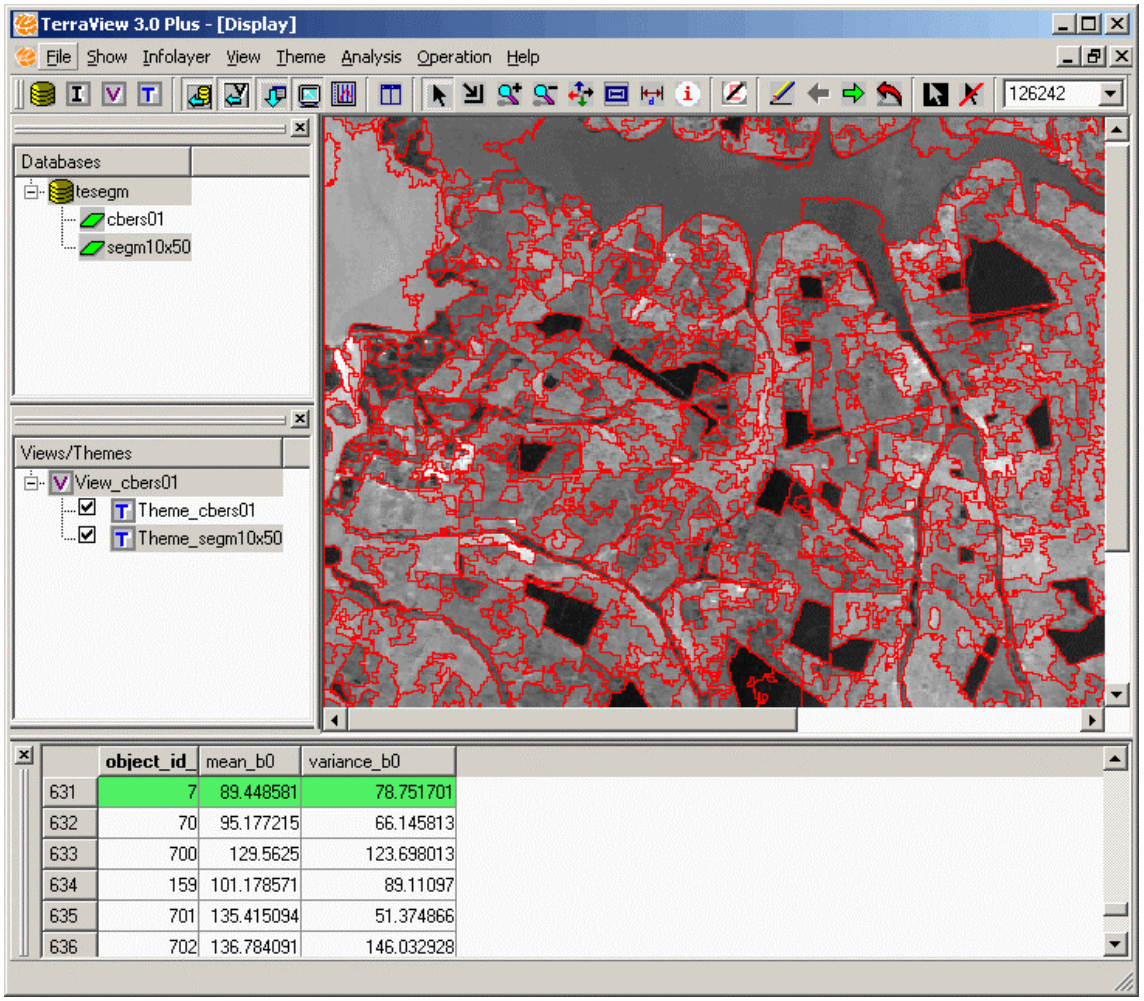

Figura 5.17 - Resultado da operação *getSegmentation* visualizado no TerraView

CAPÍTULO 6

CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou a especificação e implementação de um Serviço *Web* para Segmentação de Imagens (WISS) baseado no arcabouço de serviços proposto pelo OGC, chamado de *OpenGIS Services Framework*.

Com relação aos objetivos propostos, o protótipo desenvolvido com base na especificação WISS quando acoplado a bases de dados dos projetos PRODES e/ou DETER, permitirá aos usuários utilizarem as imagens de sensoriamento remoto desses projetos, possibilitando executar o procedimento de segmentação sobre uma ou mais imagens selecionadas deste repositório, ou de utilizar este mesmo procedimento sobre outras imagens disponíveis em outros repositórios ou fornecidas pelo próprio usuário do serviço.

O reuso das imagens armazenadas nos grandes bancos de imagens reduz tempo e custos na sua aquisição. Neste sentido, este trabalho dá sua contribuição, ao apresentar uma especificação que pode ser acoplada com facilidade aos sistemas e processos que demandam ou geram dados e informações, existentes ou em desenvolvimento nas organizações utilizando protocolos padrões e abertos, com uma combinação de artefatos criados por instituições conceituadas, como os consórcios OGC e W3C. Aumentando assim a atual capacidade de interpretar e analisar estes dados, reduzindo custos e agilizando processos de decisão.

Com relação às tecnologias utilizadas na implementação do serviço WISS, a escolha da aplicação *MapServer* não foi a melhor opção. O *MapServer* é uma aplicação desenvolvida na linguagem de programação C que utiliza o protocolo de comunicação com a Internet, o *Common Gateway Interface* – CGI e foi utilizado no protótipo desenvolvido como *front-end* do serviço WISS para receber as solicitações via *Web*.

A melhor alternativa seria utilizar a linguagem de programação JAVA que quando comparada a linguagem de programação C oferece muito mais recurso para o desenvolvimento de aplicações que são executadas no servidor *Web*. A interface com a biblioteca Terralilb onde estão disponiveis os algoritmos de segmentação pode ser feita através de *Java Native Interface* – JNI recurso que possibilita a interface entre Java e um código nativo escrito em C/C++ ou *assembly*.

Outro ponto a se discutir é o uso das especificações de serviços *Web* do OGC que embora sejam especificações voltadas para serviços geográficos para Web não foi encontrada na literatura nenhuma referencia para algumas restrições citadas mais adiante, no que diz respeito às limitações do protocolo HTTP ao período de conexão com um servidor *Web*. Uma possível alternativa para superar esta restrição seria utilizar o protocolo SOAP conforme descrito por (Alameh, 2003) e o OGC não da suporte ao protocolo SOAP nas suas especificações.

Abaixo são citadas algumas restrições do serviço, que poderão ser exploradas em trabalhos futuros:

1) Os serviços de registro ou catálogo especificados atualmente pelos consórcios OGC e W3C, como UDDI do W3C e os *Web Registry Service* (WRS) e *OpenGIS Catalog Service* do OGC não suportam nenhum tipo de consulta/restrição espacial nos seus catálogos de serviços publicado. Não poder informar a área geográfica desejada para procurar por serviços ou por dados constitui uma limitação real para os usuários. Remanesce ser visto se as versões futuras desses serviços suportarão tal funcionalidade.

2) O processo de segmentação implementado atualmente na TerraLib não permite o processamento de uma imagem digital por seleção de uma área específica. Esta opção é interessante quando o usuário está realizando testes para encontrar o melhor valor para os parâmetros de similaridade e de área, esses valores são muito importantes no contexto do algoritmo de segmentação.

3) Vários processos no tratamento digital de imagens demandam certo tempo para sua conclusão. O protocolo HTTP por ser *stateless*, e o período de conexão (*time-out*) com um servidor *Web* que normalmente expira antes de 90 segundos, são fatores que acabam limitando o tempo de processamento dessas imagens. Uma alternativa para superar esta restrição é adicionar ao cabeçalho do protocolo SOAP informações que permitam especificar o destino final de uma mensagem de serviço *Web*, com o término do processamento de uma imagem, o serviço enviaria uma mensagem para um outro serviço que implementa um mecanismo para notificar clientes (ou outros serviços) quando um serviço termina uma operação (Alameh, 2003).

Ainda como trabalhos futuros, podemos citar a implementação de um *Web Image Classification Service* (WICS) para recuperar/ler a imagem rotulada resultante do serviço WISS e executar um processo de classificação de imagens por regiões (Bins et al., 1993).

REFERÊNCIAS BIBLIOGRÁFICAS

Alameh N. Chaining Geographic Information Web Services. **IEEE Computer Society,** v. 7, n. 5, p. 22-29, Sept-Oct 2003.

Anderson G.; Moreno-Sanchez R. Building Web-Based Spatial Information Solutions around Open Specifications and Open Source Software. **Transactions in GIS.** vl. 7, n. 4, p. 447, Mar 2003.

Beaujardiere J. **OpenGIS Web Map Service (WMS) implementation specification**. [S.l.]: Open Geospatial Consortium, Inc., 2004. Document number OGC 04-024 Version: 1.3.

Bins L.S.; Erthal G.J.; Fonseca L.M.G. Satellite Imagery Segmentation: a region growing approach. In: Simpósio Brasileiro de Sensoriamento Remoto, 8., 1996, Salvador. **Anais...** São José dos Campos: INPE, abr. 1996.

Bins L.S.; Erthal G.J.; Fonseca L.M.G. Um método de classificação não-supervisionada por regiões. In: Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens, SIBGRAPI VI, 6. 1993., Recife, PE. **Anais...** Recife: [S.n], p.65-68.

Boucelma O.; Lacroix Z.E.M. WFS-Based Mediation System for GIS Interoperability. In: ACM international Symposium on Advances in Geographic Information Systems, 10., 2002, McLean, VA, USA. **Proceedings...** MacLean: ACM, 2002. p. 23-28.

BUSQUIM E SILVA, R. A. **Interoperabilidade na representação de dados geográficos**: GeoBR e GML 3.0 no contexto da realidade dos dados geográficos no Brasil. 2003-09-29. 148 p. (INPE-10295-TDI/914). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos. 2003. Disponível na biblioteca digital UR*Lib*: <http://mtc-m16.sid.inpe.br/rep-/sid.inpe.br/jeferson/2003/12.03.09.28>. Acesso em: 13 dez. 2006.

Câmara G.; Freitas U.M.; Souza R.C.M.; Garrido J. SPRING: integrating remote sensing and GIS by object-oriented data modelling. **Computers and Graphics**, v. 15, n.6, jul 1996.

Câmara G.; Souza R.C.M.; Pedrosa B.; Vinhas L.; Monteiro A.M.; Paiva J.A.; Gattas M. TerraLib: technology in support of GIS innovation. II Workshop Brasileiro de Geoinformática. GeoInfo2000. São Paulo. **Anais...** São Paulo: SBC, 2000.

Casanova M.A.; Câmara G.; Davis C.; Vinhas L.; Queiroz G. **Bancos de dados geográficos**. Curitiba: MundoGEO, 2005. 490p.

Cox S.; Daisey P.; Lake R.; Portele C.; Whiteside A.; **OpenGIS® Geography Markup Language (GML) implementation specification**. [S.l]: Open Geospatial Consortium, Inc. 2003.

Deng M.; Liu Y.; Di L., An interoperable web-based classification service for remote sensing data. In: ASIA GIS 2006 International Conference, 6., 2006, Oct 16-18, Skuda, Malaysia.. **Proceedings…** Skuda, Malaysia: UTM, 2006.

Deng M.; Zhao P.; Liu Y.; Chen A.; Di L, The Development of A Prototype Geospatial Web Service System for Remote Sensing Data. International Society for Photogrammetry and Remote Sensing (ISPRS)**,** 20., 2004 - Istanbul, Turkey.. **Proceedings…** Istanbul: ISPRS, 2004, p. 218.

Espindola G.; Câmara G.; Reis I.; Bins L.; Monteiro A.M.; Spatial Autocorrelation Indicators for Evaluation of Remote Sensing Image Segmentation Algorithms. In: Annual Conference of the International Association for Mathematical Geology (IAMG), 2005, Ago 22, Toronto, Canada. **Proceedings...** Toronto:IAMG, 2005.

Espindola G.; Crusco N.A.; Epiphanio J.C.N.; Aplicação da metodologia do PRODES digital em imagens CCD/CBERS-2. In: Simpósio Brasileiro de Sensoriamento Remoto, 12., 2005, Goiânia. **Anais…** São José dos Campos: INPE, 2005.

ESRI ArcView image analysis extension. In: ESRI European User Conference (EEUC), 1988, Firenzi. **Proceedings…** Firenze, Italy, ESRI, 1998.

Essid M.; Boucelma O.; Colonna F.M.; Lassoued Y.; Query Processing in a Geographic Mediation System. In: Annual ACM International Workshop on Geographic Information Systems, 12., 2004, New York. **Proceedings…** Washington DC, USA: ACM, 2004. p. 101-108.

Extensible Markup Language (XML). Disponível em: <http://www.w3.org/XML/>, Acesso em 05 dez. 2005.

Extensible Markup Language-Remote Procedure Calling (XML-RPC). **Remote procedure calling protocol**. Disponível em: <http://www.xmlrpc.com/>, Acesso em 19 out. 2005.

Freitas U.; Monteiro A.; Câmara G.; Souza R.; Ii F. Development of an Integrated Image Processing and GIS Software for the Remote Sensing Communit. In: Annual Conference of the Remote Sensing Society, Reading, 23., 1997, Reading. **Proceedings…** Reading: RSS97, p.237-242, 1997.

Furlan J.D. **Modelagem de objetos através de UML**. São Paulo: Makron Books, 1998. 329p.

Gardels K. The Open GIS Approach to Distributed Geodata and Geoprocessing. In: International Conference/Workshop on Integrating GIS and Environmental Modeling, 3., 1996, Santa Fe, NM, USA. **Proceedings…** Santa Fe: NCGIA, 1996, p. 21-25,

Gonzales R.C.; Wintz P. **Digital image processing**. Proding: Addison – Wesley, 1987, 431p.

Intergraph Corporation. GeoMedia Image. Disponível em: <http://www.intergraph.com/gimage/>, Acesso em 20 jan. 2006.

Internet Inter-ORB Protocol (IIOP). Disponível em: <http://www.omg.org/>, Acesso em 11 ago. 2005.

Kottman C. **OpenGIS catalog services**. [S.l]: Open Geospatial Consortium, Inc. 1999. Document number OGC 99-113 Version: 4.3

MapServer. Disponível em: <http://mapserver.gis.umn.edu/>, Acesso em 09 jan. 2005.

Mascarenhas N.D.; Velasco F. **Processamento digital de imagens**. Buenos Aires: Kapelusz, 1989. 272p.

Mather P. M.; **Computer processing of remotely-sensed images:** an introduction. New York: John Wileu & Sons, 1999.

Microsoft. **Distributed Component Object Model (DCOM)**. Disponível em: <http://www.microsoft.com/com/>, Acesso em 11 ago. 2005.

Miranda J.I.; **Publicando mapas na Web:** servlets, applets ou CGI?. Campinas: Embrapa Informática Agropecuária , 2003. 38 p.

NASA. **Earth science enterprise scientific data purchase program**. Disponível em: <https://zulu.ssc.nasa.gov/mrsid/>, Acesso em 01 fev. 2006.

NASA/JPL. **OnEarth.** Disponível em: <http://onearth.jpl.nasa.gov/> - Acesso em 27 jan. 2006.

OpenGIS Consortium Inc. Disponível em: <http://www.opengeospatial.org/>, Acesso em 27 jan. 2006.

PCI Geomatics Corporation. **PCI geomatics**. Disponível em: <http://www.pcigeomatics.com/>, Acesso em 20 jan. 2006.

Pekkarinen A. A method for the segmentation of very high spatial resolution images of forested landscapes. **International Journal of Remote Sensing**, v. 23, n. 14, p. 2817-2836, July, 2002.

Peng Z.R.; Tsou M.H. **Internet GIS:** distributes deographic information services for the Internet and Wireless Networks. New Jersey: John Wiley & Sons, Inc. 2003.

Percivall G. **OpenGIS® reference model**. [S.l.]: Open Geospatial Consortium, Inc. 2003. Document number OGC 03-040 Version: 0.1.3.

Portable Network Graphics (PNG). Disponível em: <http://www.w3.org/Graphics/PNG/>, Acesso em 20 jan. 2006.

Remote Method Invocation (RMI). **Java remote method invocation**. Disponível em: <http://java.sun.com/products/jdk/rmi/>, Acesso em 11 ago. 2005.

Rumbaugh J.; Booch G.; Jacobson I. **UML** guia do usuário. [S.l.]: Editora: Campus, 2000. 472p.

Rumbaugh. J.; Booch G. Jacobson I. **The unified software development process**.: Pearson: Addison-Wesley Professional, 1999. 463p.

Scalable Vector Graphics (SVG). Disponível em: <http://www.w3.org/Graphics/SVG/>, Acesso em 20 jan. 2006.

Shimabukuro Y.E.; Smith J.A.; The least-squares mixing models to generate fraction images derived from remote sensing multispectral data. **I.E.E.E. Transactions on Geoscience and Remote Sensing**, v. 29, n., p.16-20, 1999

Shimabukuro Y.E.; Valeriano D.M.; Duarte V.; Detecção do desflorestamento da Amazônia Legal em tempo real - Projeto DETER. In: Simpósio Brasileiro de Sensoriamento Remoto, 12., 2005, Goiânia, Brasil. **Anais...**São José dos Campos: INPE, p. 3403-3409.

Silva M. P. S. **Mineração de imagens usando ontologias**. Disponível em: <http://www.dpi.inpe.br/~mpss/>, Acesso em 27 jan. 2006.

Simple Object Access Protocol (SOAP). Disponível em: <http://www.w3.org/TR/soap12/>, Acesso em 05 dez. 2005.

Sonnet J. **OWS 2 Common Architecture:** WSDL SOAP UDDI. Local: [S.l.], Open Geospatial Consortium, Inc. 2005. Discussion Paper OGC 04-060r1, Version: 1.0.0
Terry L. Wireless GIS – Concept & Reality. In: Symposium on Geospatial Theory, Processing and Aplications, 2002, Ottawa. **Proceedings…** Ottawua: ISPRS, 2002.

Unified Modeling Language (UML) 2.0. Disponível em: <http://www.uml.org/>, Acesso em 15 set. 2005.

Valeriano D.M.; Mello E.M.K.; Moreira J.C.; Shimabukuro Y.E.; Duarte V.; Souza I.M.; Santos J.R.; Barbosa C.C.F.; Souza R.C.M. Monitoring tropical forest from space: the prodes digital project. In:International Society for Photogrammetry and Remote Sensing Congress, 20, 2004, Istanbul. **Proceedings…** Istanbul: ISPRS, 2004.

Vretanos P. **OpenGIS Web Feature Service (WFS) implementation specification.** [S.l.]: Open Geospatial Consortium, Inc. 2005. Document number OGC 04-094 Version: 1.1.

Web Services Architecture. Disponível em: <http://www.w3.org/TR/ws-arch/>, Acesso em 05 dez. 2005.

Web Services Description Language (WSDL). Disponível em: <http://www.w3.org/TR/wsdl20/>, Acesso em 05 dez. 2005.

Whiteside A. **OpenGIS web service common** implementation specification. [S.l.]: Open Geospatial Consortium, Inc. 2005. Document number OGC 05-008c1 Version: 1.0.

World Wide Web Consortium. Disponível em: <http://www.w3.org/>, Acesso em 27 jan. 2006.

World Wide Web Consortium: **Web services**. Disponível em: <http://www.w3.org/2002/ws/>, Acesso em 10 jan. 2006.

APÊNDICE A

INTEGRAÇÃO TERRALIB X MAPSERVER

Neste anexo é apresentado uma descrição completa da integração da biblioteca TerraLib e o servidor de Mapas MapServer, esta descrição inclui a metodologia utilizada, as funcionalidades desenvolvidas e a aplicação de teste implementada.

A.1 Metodologia

A integração baseou-se na inclusão de um novo arquivo de programa no servidor de mapas MapServer, escrito em linguagem C/C++, o uso dessas duas linguagens foram necessárias devido ao código fonte do MapServer estar desenvolvido na linguagem C e a biblioteca Terralib estar em C++. Para que esta mistura de código C/C++ tenha sucesso, existem alguns pontos que devem ser observados.

1) É preciso usar o compilador C++ para compilar o *main()* da aplicação, por exemplo, para inicialização estática;

2) O compilador C++ deve dirigir o processo de *linking**, para que as bibliotecas possam ser resgatadas;

3) Os compiladores C e C++ devem ser adquiridos do mesmo fornecedor, e devem ser de versões compatíveis, por exemplo, devem usar a mesma convenção de chamadas a funções.

(*) *linking*: Um programa que combina os resultados da compilação de um ou de vários arquivos em um único arquivo executável pelo computador. *Linkage specification* é uma notação que diz ao compilador C++ que função foi, ou deve ser, compilada com as convenções de ligação de uma outra linguagem.

Para que as funções criadas em C++ possam ser chamadas a partir de código C, o compilador C++ precisa saber que as funções serão chamadas por um compilador C, para que isso ocorra é preciso indicar para o compilador usando a construção `extern "C"`.

A construção `extern "C"` indica ao compilador que a informação externa enviada ao *linker*, deve usar uma convenção de chamada e *name mangling*[*] padrão C, por exemplo, precedida por um caractere *underscore*. Uma vez que a sobrecarga de nomes não é suportada pelo C, não é possível fazer várias sobrecargas de função que possam ser chamadas simultaneamente por um programa C.

(*) *name mangling*: é uma técnica usada pelo compilador C++ para garantir ligação segura. Essa técnica reformula internamente os nomes das funções, de tal modo que os nomes passam a refletir não apenas a função, mas também sua lista de parâmetros com os correspondentes tipos de dados.

Para testar a integração dos dois ambientes, foi desenvolvida uma interface *Web* de consulta geográfica sobre o *MapServer*. Essa interface tem como objetivo utilizar as operações implementadas e mostrar graficamente os resultados na área de visualização do mapa da interface, facilitando assim os testes.

A.2 Funcionalidades Desenvolvidas

Estas funcionalidades foram implementadas em dois níveis:

1) No primeiro nível as operações fazem interface com o servidor de mapas MapServer, qualquer requisição enviada ao servidor destinada ao SGBD TerraLib, são recebidas por estas operações que posteriormente acionam as operações implementadas no segundo nível onde são utilizadas as classes da biblioteca TerraLib. Os parâmetros recebidos por estas operações são: parâmetros de conexão para um determinado SGDB, o nome do *layer* onde estão armazenadas as informações vetoriais ou matriciais, o tipo de geometria (ponto,

linha e polígono) e os parâmetros para consulta e visualização, como o retângulo envolvente ou o identificador único de uma geometria. Em resposta as solicitações de dados vetoriais, são instanciadas os objetos da classe `shapeOBJ` do MapServer com as informações recuperadas. Nesta versão atual do protótipo os dados matriciais são utilizados apenas pelo serviço de segmentação de imagens, não estando disponíveis para visualização no MapServer.

2) As operações do segundo nível fazem interface com a TerraLib sendo responsável pela conexão com o SGDB, recebendo os parâmetros do nível acima essas operações recuperam as informações vetoriais requisitas pelo servidor de mapas através de SQL criados dinamicamente, convertendo as informações para uma estrutura de dados geográficos do MapServer. Também estão implementadas neste nível as operações do WISS, onde são processadas as imagens de sensoriamento remoto armazenadas em disco no formato *GeoTiff* ou no SGDB, com os algoritmos de segmentação, o resultado do processo é disponibilizado no formato *Shapefile* podendo ser acessado por *download* do arquivo ou convertido para o formato GML utilizando a operação *GetFeature* do serviço WISS.

As operações implementadas no nível 2 utilizaram as classes da TerraLib, quando uma conexão com SGDB é solicitada pelo servidor de mapa, é instanciado então, um objeto da classe genérica `TeDatabase`, este objeto é instanciado de acordo com o SGDB solicitado podendo ser MySQL, PostGreSQL, Oracle ou Access, esta operação segue três etapas:

1) Leitura dos parâmetros de conexão com o SGDB e o nome do *layer* referente ao dado vetorial que esta sendo requisitado.

2) Recuperação das geometrias do SGBD através de expressões SQL: o SGBD retorna as geometrias como conjuntos de valores binários (BLOB), o `TeDatabase` faz a leitura de cada BLOB e transforma para uma das estruturas

de dados geográficos da *TerraLib* (`TePolygon`, `TeLine2D`, `TePoint`, etc).

3) Conversão da estrutura de dados *TerraLib* para *MapServer*.

A.3 Protótipo do Sistema

Para testar a integração da TerraLib com o Mapserver um protótipo do sistema foi desenvolvido, a Figura A.1 ilustra a integração no contexto de uma aplicação *Web*, *TerraLib/MapServer* e SGBD. As páginas *Web* desenvolvidos sobre o *MapServer* utilizam as novas funcionalidades para realizarem operações espaciais sobre os dados geográficos armazenados nos SGBD. Essas funcionalidades são responsáveis pela interface com a *TerraLib* e pela interpretação dessas operações.

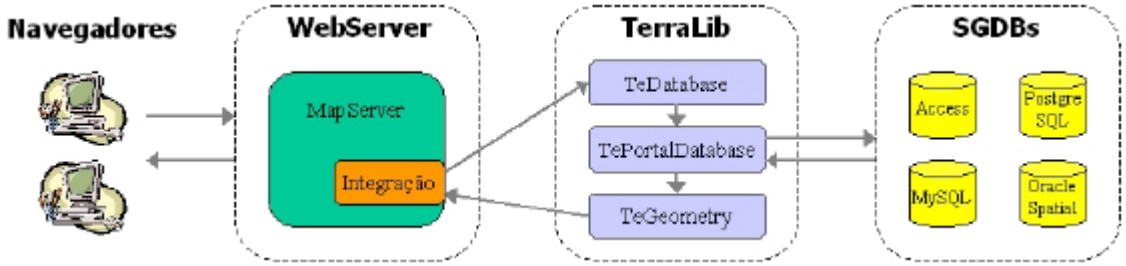

Figura A.1 - Integração TerraLib/MapServer

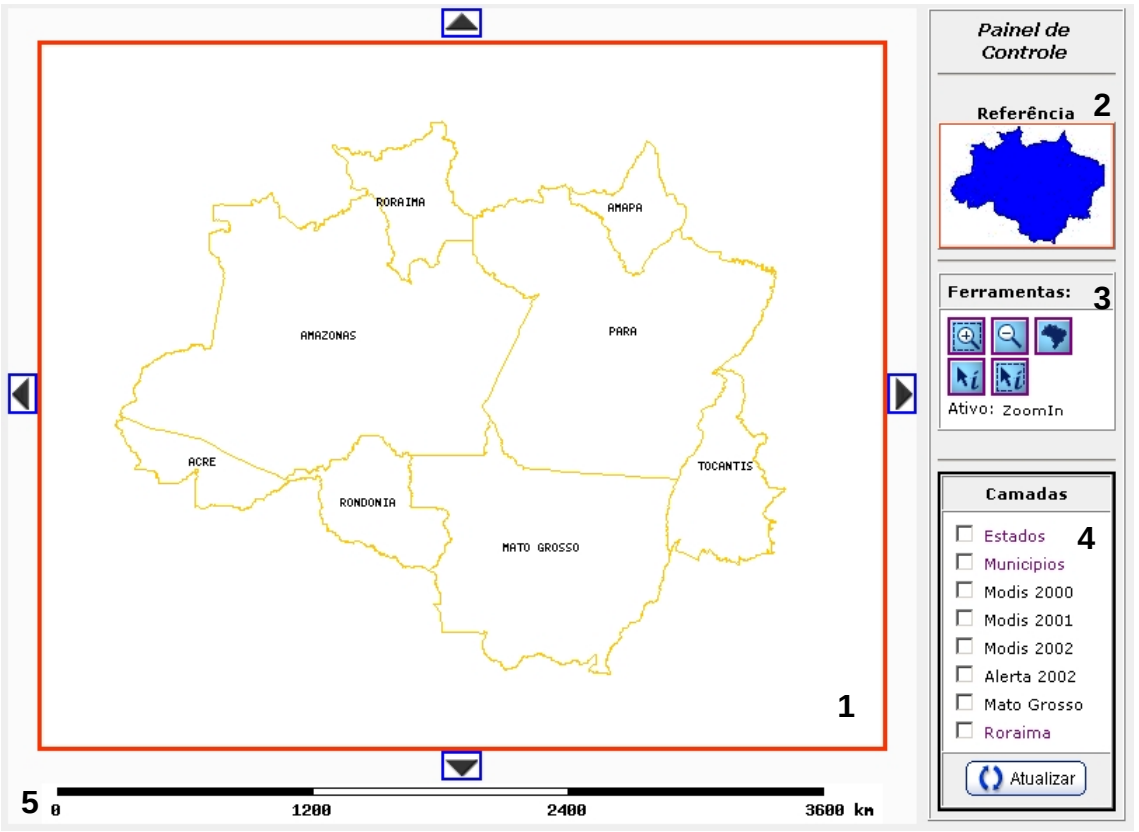

Figura A.2 - Tela de navegação da aplicação MapServer

1) Área reservada para visualização dos dados geográficos, onde o usuário poderá interagir com o mapa selecionando as opções descritas nos itens (2), (3) e (4).

2) *OverView* é usada pelo usuário para se deslocar mais rapidamente de um local para o outro, onde através de um único clique é possível visualizar uma outra extremidade do mapa independente do zoom.

3) A barra de ferramentas disponibiliza ao usuário as funções de *Zoom In*, *Zoom Out*, Recompor, Consultar a Camada superior (*Info*) ou Consultar todas as camadas (*nInfo*) ativas no item (4).

4) As 'Camadas' possibilitam ao usuário ativar ou desativar todos os mapas disponíveis no *Web Site*, a ordem de prioridade de visualização dos mapas em relação aos outros é seguida pela ordem colocada no *Web Site*, isto é, o primeiro mapa (Estados) é o que tem menos prioridade de visualização.

5) A escala gráfica permite ao usuário ter uma idéia da dimensão do mapa visualizado em um determinado momento, estndo relacionada diretamente a função de *zoom*.

O código apresentado abaixo mostra um exemplo de como os desenvolvedores deverão configurar o arquivo *Mapfile* do *MapServer*, para que os dados espaciais disponíveis nos SGDB possam ser acessados.

```
LAYER
    NAME DivisaoEstadual

    CONNECTIONTYPE terralib
    CONNECTION "driver=mysql user=root password=*** dbname=mydb host=luigi-pc port=3306"
    DATA "estados"
    STATUS OFF
    TYPE POLYGON

    CLASS
       COLOR 255 0 0
    END
END
```