Leonardo Lacerda Alves

Avaliação de Serviços Web do Open Geospatial Consortium para Infra-estruturas de Dados Espaciais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Informática.

Belo Horizonte

Fevereiro de 2007

Resumo

Interoperabilidade é um fator essencial para os sistemas de informação geográficos (SIG). Nos últimos anos, pesquisas e esforços para suportar interoperabilidade evoluíram da troca *offline* de arquivos padronizados para a consolidação dos repositórios *online* de dados espaciais, levando às primeiras iniciativas de tratamento de aspectos semânticos de dados geoespaciais. O *Open Geospatial Consortium* (OGC) tem proposto um conjunto de padrões com a intenção de promover interoperabilidade através do uso de serviços. Entretanto, questões relacionadas com tolerância a falhas, implementação independente de provedores, suporte a transações demoradas e assíncronas, privacidade e outras refletem a necessidade de estudos e discussões adicionais. Esta pesquisa concentra-se na implementação de interoperabilidade através de serviços e no papel das arquiteturas orientadas a serviços para a implementação e difusão das infra-estruturas de dados espaciais urbanas (LSDI). É apresentado um protótipo, o qual foi projetado para testar o modelo abstrato de serviços e informação geográfica através da implementação de um caso de uso real. Como conclusão, são indicados melhoramentos que mostram-se úteis para o desenvolvimento de clientes de SIG e LSDI, para a comunicação entre servidores e clientes magros, além de capazes de habilitar clientes a suportarem mecanismos de tolerância a falhas sem dependência de provedores.

Abstract

Interoperability is one of the most important challenges related to GIS. Through the last years, research on interoperability has evolved from the simple off-line exchange of standardized-format files, through the establishment of spatial data clearinghouses, and to the first initiatives in the treatment of semantic aspects of data. The Open Geospatial Consortium (OGC) has proposed a number of standards to that respect, with the intention of promoting interoperability through the use of services. However, issues regarding fault tolerance, server-independent implementation, delayed-time transactions, privacy, and others reflect the need for further study and discussion. This work discusses the current status of service-oriented architectures as applied to interoperable GIS, or, more specifically, to the implementation of local spatial data infrastructures (LSDI). A prototype designed to test the services abstract model by simulating a real-world use case is presented. Our conclusions indicate that some improvements may be helpful in the development of GIS/LSDI clients, as well as in the communication between thin clients and servers, supporting delayed-time transactions through asynchronous communication, and enabling clients to support fault-tolerant mechanisms without provider-dependent solutions.

Mi dediĉas ĉi tiun laboron por
Gepatroj kaj Fernanda
tiuj kiuj helpas min tra la vera vojo

Agradecimentos

A participação neste curso e a realização dos muitos trabalhos que esta dissertação sintetiza deu-se com a ajuda de muitos, sem os quais meu esforço seria maior e mais difícil. Logo, deixo registrado meus agradecimentos:

- aos muitos deuses que colaboraram com este trabalho, em especial agradeço ao meu Pai por ter aguardado com paciência que eu concluísse tais atividades.
- à minha mãe, ao meu pai, meu irmão Leonel e minha noiva Fernanda, os quais doaram seu tempo e se abdicaram do meu tempo para eles, incondicionalmente.
- ao meu orientador, Prof. Dr. Clodoveu Augusto Davis Jr; com quem pude aprender pelo exemplo de responsabilidade, prontidão e diligência.
- à turma do doce de leite: Michelle, Anne, José Geraldo, Cristiano, Pedro, Sandro e Wanderley. E aos colegas Fabrício, Gustavo, Sebastião, Kátia, Caroline, Cláudio, Marcelo, Uirá, Reinaldo, Carlos e Fábio.
- àqueles que foram meus professores durante o curso: Mark Alan, Ana Maria, Lucila e Sílvio Jamil. E àqueles que contribuíram com inspiradoras conversas: Marcos André Gonçalves, Ricardo Poley e Marco Túlio.
- à Giovana e à Priscila, da secretaria acadêmica, por terem sido gentis e prontas para nos ajudar.
- ao Pedro Felipe que participou dos nossos experimentos.
- ao pessoal do setor de informática por terem dado suporte aos nossos trabalhos: Marco Aurélio, Leonardo Vilela, Bruno, Emanuel e Marcus Vinícius.
- às instituições que financiaram o meu curso: A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), a Pontifícia Universidade Católica de Minas Gerais (PUC-MG) e a Fundação Comunitária de Ensino Superior de Itabira (Funcesi).

- à banca da defesa de dissertação, a qual contribuiu para melhorar o meu trabalho e foi constituída pelos professores: José Luís Braga, Marco Túlio de Oliveira Valente e Clodoveu Augusto Davis Jr.

- aos *referees* anônimos dos eventos Geoinfo 2005, Geoinfo 2006 e ACM GIS 2006, pelas críticas essenciais.

- às demais pessoas que torceram por mim.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Sistemas de Informação Geográficos (SIG) são sistemas para coleta, edição, armazenamento, análise, gerenciamento e integração de dados com atributos associados a alguma localização na Terra. São constituídos basicamente por equipamentos (*hardware*), programas de computador (*software*), técnicas e pessoas, os quais são usados como instrumento de apoio a decisão em governos, empresas, pesquisas científicas e, atualmente, até mesmo em atividades pessoais ou domésticas [CCD+05].

A tecnologia de SIG beneficia inúmeras aplicações, que passam pelo controle de tráfego em cidades e rodovias, redução do custo de logística de produtos, sistemas de roteamento em veículos particulares, até o controle efetivo de áreas de plantio e preservação ambiental. Além destas, SIG têm sido usados também para estimular a participação de cidadãos comuns na administração municipal [Gho01].

1.1 Motivação

Como sistemas de computação, SIG são acessíveis através de aparelhos comuns tais como computadores pessoais, celulares e PDAs (*Personal Digital Assistants*), bem como através de dispositivos específicos, tais como navegadores GPS[1], os quais têm se tornado populares em países desenvolvidos e em grandes centros urbanos. O acesso a esses sistemas através de computadores pessoais tem se popularizado através de softwares cada vez mais amigáveis, como o Google Earth, e sites Web tais como Google Maps[2], Yahoo Maps[3] e MapLink[4].

[1]GPS é sigla de *Global Positioning System*, um sistema de localização mundial que usa referências de uma constelação de satélites artificiais para determinar posições na superfície terrestre.

[2]URL: maps.google.com/

[3]URL: maps.yahoo.com/

[4]URL: maplink.uol.com.br/

De fato, aplicações de SIG são crescentes na medida em que dados geográficos tornam-se disponíveis e novas tecnologias são difundidas para melhorar a forma de aquisição, processamento, análise e saída de informação geográfica [AD07]. Conseqüentemente, as dificuldades atuais para implementação, gerenciamento, distribuição e uso amplos de SIG não favorecem a disponibilidade de informação e aplicações e portanto são lacunas importantes para pesquisas na área de geoinformática.

1.2 Justificativa

Apesar da significativa oferta de informação geográfica atualmente, boa parte dos fenômenos de interesse para as aplicações é dinâmica, ou seja, mudam com tal freqüência que seus dados precisam ser constantemente atualizados e coletados novamente. Isso ocorre com especial ênfase em espaços urbanos, os quais crescem continuamente, têm sua organização física alterada e sofrem a influência de diversas ações causadoras de mudanças, tais como a legislação de uso e ocupação do solo, as regras de circulação de veículos, a distribuição de atividades econômicas e outras. É evidente que a atualização de dados sobre fenômenos com tais características é algo trabalhoso e custoso.

Freqüentemente, a necessidade de informação desse tipo é compartilhada por mais de uma organização, o que possibilita o compartilhamento dos dados coletados e minimiza custos relativos a atividades redundantes de coleta, tratamento do dado geoespacial e até mesmo seu armazenamento [DA05]. No entanto, esse potencial para a colaboração por meio do compartilhamento de dados é muitas vezes prejudicado pela existência de diferentes formas de modelar o raciocínio geográfico sobre o mesmo espaço, ou seja, um indivíduo pode modelar residências como pontos, enquanto outro as define como polígonos. Essa aparente incompatibilidade entre alternativas de representação poderia ser resolvida através de iniciativas de padronização em que a representação mais detalhada fosse adotada. Em muitos casos, a conversão de uma representação com mais detalhes para outra com menos detalhes é possível, mas o contrário não é igualmente possível [DL99]. Assim, uma vez que a representação computacional de um objeto ou fenômeno é uma simplificação do fenômeno geográfico real, certos detalhes são perdidos na representação e não podem ser inferidos a partir do dado geográfico já coletado, o que aumenta a responsabilidade da coordenação de esforços de cooperação entre instituições para garantir que um dado geográfico compartilhado seja útil para todos os participantes. Por outro lado, não é viável fazer com que todo dado seja coletado com um nível de detalhamento muito grande, pelo alto custo de implementação, dificuldades de manutenção e pelo fato de não ser simples antecipar necessidades futuras dos usuários.

Neste contexto, é do interesse de todos os participantes de uma iniciativa de compartilhamento de dados geográficos que exista compatibilidade tecnológica e semântica entre os dados provenientes das diferentes instituições envolvidas. A parcela tecnológica da questão, apesar de ter sido mais intensivamente estudada, ainda causa muitos problemas, pois as instituições usam diferentes softwares, tecnologias de bancos de dados e sistemas de referência geográfica, diferenças estas sempre muito comuns. A parcela semântica ainda está sendo investigada, não existindo até o momento uma solução definitiva [FM05]. A essa capacidade de funcionamento interdependente, colaborativo e com trocas transparentes de dados denominamos *interoperabilidade*.

Uma das iniciativas mais recentes de promover interoperabilidade foi coordenada pelo Open Geospatial Consortium (OGC), o qual tem proposto padrões para o uso de serviços no compartilhamento de informação geográfica [Per03]. No entanto, a definição e especificação original do OGC de serviços para acesso a informação geoespacial antecedem e diferem da definição e especificação de serviços Web desenvolvida pelo *World Wide Web Consortium* (W3C) [Whi05]. Logo, alcançar a interoperabilidade efetiva entre SIG e outros sistemas não geográficos por meio do uso de serviços em rede depende de ajustes entre as propostas do OGC e do W3C, os quais estão em andamento [Son04].

Enquanto esses ajustes não são completados, observam-se dificuldades quando se busca implementar interoperabilidade através de serviços. Questões tais como baixa tolerância a falhas, dependência de provedores de serviços geográficos, dificuldade de execução de tarefas demoradas, ausência de parâmetros de privacidade e outras refletem a necessidade de mais estudos e discussões.

A existência dos padrões OGC indica um grande potencial para o desenvolvimento de sistemas de informação geográficos de perfil bem diferente dos atuais sistemas monolíticos. Torna-se possível conceber SIG interoperáveis e distribuídos em rede, constituídos de módulos altamente especializados e fracamente acoplados. No entanto, são escassos os estudos sobre o desenvolvimento de sistemas distribuídos em conformidade com os padrões da OGC [SKK06]. Apesar da significativa disponibilidade de serviços OGC para fins de distribuição de dados de escala nacional ou regional em muitos países do mundo [Ref06], e da correspondente – ou latente – demanda de clientes para esses serviços, não se conhece a disponibilidade de serviços específicos para regiões geograficamente menores, tais como estados e cidades.

1.3 Objetivo

Esta pesquisa concentra-se na implementação de interoperabilidade através de serviços e no papel das arquiteturas orientadas a serviços para a implementação e difusão das infra-estruturas de dados espaciais urbanas (LSDI, do inglês *Local Spatial Data Infrastructures*).

A maioria das infra-estruturas de dados espaciais (SDI, do inglês *Spatial Data Infrastructures*) atualmente conhecidas, como, por exemplo, o projeto "*Infrastructure for Spatial Information in Europe*" (INSPIRE) da Comunidade Européia e o *National Spatial Data Infrastructure* (NSDI), dos Estados Unidos, referem-se a dados de escopo continental ou nacional, mas as LSDI contêm conjuntos potencialmente mais complexos e detalhados de dados geográficos [NBFRW04, RWHJ00, AD06], e portanto envolvem um número maior de tipos de serviços, os quais podem ser usados através de dispositivos diversos, tais como PDAs, telefones celulares e computadores pessoais.

Assim, o objetivo deste trabalho é investigar a adequação dos padrões OGC para a implementação de (1) sistemas de informação geográficos baseados em dados compartilhados em rede, (2) infra-estruturas de dados espaciais de âmbito local, e (3) softwares-cliente para tais SIG e LSDI.

Os objetivos específicos deste trabalho incluem:

- Implementar um protótipo de SDI e SIG interoperáveis, com informações de uma região metropolitana, de acordo com as especificações da OGC;
- Avaliar a adequação da arquitetura orientada a serviços proposta pela OGC para a implementação de serviços geográficos no contexto de aplicações urbanas reais;
- Investigar se há requisitos da OGC que não podem ser facilmente atendidos no desenvolvimento de softwares-cliente, considerando o interesse de manter tais clientes independentes da SDI desenvolvida e de dispor de clientes destinados para diferentes tipos de dispositivos de acesso a serviços geográficos, tais como PDAs, celulares e computadores pessoais comuns.

1.4 Metodologia

A avaliação de adequação e conformidade dos padrões OGC com as tecnologias de desenvolvimento de softwares-cliente de SIG para diferentes LSDI deu-se por meio de uma prova de conceito, construída em quatro fases interdependentes:

1. Revisão da literatura

2. Definição de um cenário de implementação
   Foi definido um cenário baseado em um contexto urbano, com dados reais da cidade de Belo Horizonte, Minas Gerais, e com necessidades reais de usuários. Foi utilizado um cenário preliminar definido no Modelo de Referência da OGC (ORM, do inglês *OGC Reference Model*) [Per03] denominado *Consumer Travel Assistance Scenario*.

3. Desenvolvimento de Protótipo
   Foi desenvolvido um protótipo composto por serviços da arquitetura OWS (*OGC Web Services*), um provedor de serviços que integra os serviços Web da OGC e clientes para estes serviços, todos em conformidade com o modelo abstrato proposto pela OGC (*OGC Abstract Model*). O protótipo foi confrontado com os requisitos e as diferentes *perspectivas* definidas no ORM. Os clientes são dois: um que usa o provedor de serviços diretamente e os serviços Web da OGC indiretamente, e outro que usa os serviços Web da OGC diretamente, sem usar o provedor de serviços como intermediário.

4. Especialização da Arquitetura OWS (OGC Web Services)
   Para sanar as limitações identificadas, foram desenvolvidos serviços de infra-estrutura complementares e também foram implementados outros serviços da OGC, em um modelo de processo em espiral. Usando tais serviços, foi implementada uma nova versão do protótipo para assegurar que as limitações observadas foram sanadas.

1.5 Estrutura

Este trabalho está organizado em cinco capítulos.

O Capítulo 2 apresenta conceitos fundamentais sobre SIG, trabalhos relacionados à interoperabilidade entre SIG, sobre Infra-estruturas de Dados Espaciais (SDI) em geral, e Infra-estruturas de Dados Espaciais Locais (LSDI) em particular, o que inclui as idéias de implementação e configuração de tais infra-estruturas por meio de serviços Web.

O Capítulo 3 organiza os requisitos e aspectos mais importantes dos serviços Web geoespaciais especificados pela OGC e do protótipo de SIG desenvolvido. Os requisitos não-funcionais selecionados são introduzidos no desenvolvimento dos serviços de infra-estrutura e dos softwares-cliente. Finalmente, são identificadas e apresentadas as limitações dos serviços e softwares-cliente no que se refere aos requisitos não-funcionais propostos pela OGC no ORM.

No Capítulo 4, são discutidas as funcionalidades requeridas para uma LSDI que não são atendidas pelos serviços propostos no arcabouço OWS. Estas funcionalidades são apresen-

tadas através de novos serviços de infra-estrutura que resolvem as limitações identificadas no capítulo 3.

No Capítulo 5, são apresentados os principais resultados e contribuições deste trabalho, e indicadas algumas direções de pesquisas futuras.

Finalmente, são apresentadas as referências bibliográficas e o apêndice, este dividido em duas partes. A primeira parte do apêndice reúne interfaces e formatos dos serviços OGC modificados neste trabalho, enquanto a segunda parte expõe documentos de amostra, especificação de formato e regras de transformação do serviço *Transaction Control Service*.

Capítulo 2

Trabalhos Relacionados

2.1 Interoperabilidade entre SIG

Organizações com necessidade de compartilhar informação geográfica normalmente dependem de ferramentas e técnicas de tradução de dados. Isso deve-se ao fato de que cada organização potencialmente usa um software diferente para seu SIG [DA05]. No passado houve esforços para estabelecer um formato neutro de arquivo com o intuito de favorecer o intercâmbio de dados geográficos. Desta forma, a necessidade de tradução de dados entre formatos se restringiria apenas a conversões bidirecionais entre qualquer formato e aquele formato intermediário [LCdQ01].

Porém, essa técnica limita-se a aspectos sintáticos e pode apresentar dificuldades de compatibilização semântica dos dados. É possível que tais formatos neutros não tenham se popularizado exatamente por este motivo, embora um número significativo de tradutores tenham sido desenvolvidos e difundidos entre usuários de geoinformação, alguns utilizando-se de formatos intermediários enquanto outros convertendo dados diretamente entre formatos proprietários.

Adicionalmente, o uso de tradução não adaptou-se facilmente ao ambiente *online*, pois manteve a necessidade da operação *offline* (exportar-converter-importar). Conseqüentemente, problemas de sincronização tornam-se comuns, uma vez que múltiplas cópias do mesmo dado são distribuídas em momentos diferentes, com difícil controle de réplicas e alto custo de armazenamento [DA05].

De fato, apenas formatos de troca definidos por força de lei ou regulamentações governamentais foram mais usados neste processo de tradução de dados. Um exemplo é o SDTS (*Spatial Data Transfer Standard*) nos EUA [Uni98]. Mas, na prática, esse processo acontece segundo práticas difundidas na comunidade da ferramenta de SIG onde o usuário se encontra, onde alguns poucos formatos tornam-se mais comuns que outros e

onde as limitações da tradução entre formatos e as principais técnicas para saná-las são bem conhecidas [LCdQ02].

No entanto, além dos aspectos técnicos relacionados à troca de arquivos, muitos outros aspectos são importantes para estabelecer o intercâmbio efetivo de informação geográfica entre sistemas de informação vistos como sistemas sócio-técnicos. Estes aspectos incluem políticas de uso da informação e direitos sobre a mesma, procedimentos comuns de disponibilização, atualização e transferência de dados, definições sobre quais dados são básicos e como extensões são gerenciadas entre a comunidade de usuários, além de outros [DF06].

Todo este conjunto de aspectos sócio-técnicos constitui um acordo de cooperação entre usuários – instituições e pessoas – que objetivam compartilhar dados e contribuir entre si, onde especialidades e interesses definem responsabilidades sobre a atualização da informação compartilhada. Porém, o cerne desses acordos de cooperação são as políticas, procedimentos e regras de disseminação da informação entre os participantes, o que determina características sempre muito próprias para cada grupo de participantes [FCOM06].

No entanto, diferentemente das infra-estruturas de dados espaciais, as quais serão tratadas na seção 2.4, os acordos de cooperação possuem em comum a necessidade de um controle centralizado sobre as políticas ao mesmo tempo em que apresentam um controle descentralizado sobre a informação, onde cada participante tem controle sobre um grupo específico de dados, e cada qual atribui aos dados apenas aquilo que se mostra conveniente aos seus objetivos particulares. Conseqüentemente, isso exige que os usuários se adaptem às necessidades do proprietário dos dados, e não que os dados se adaptem às necessidades dos participantes [DF06].

Com o propósito de reduzir o atraso entre a produção da informação por um participante do acordo de cooperação e sua disseminação para todos os outros participantes, os repositórios e geoportais são mecanismos técnicos importantes. Mas para a necessidade de abstração da informação geográfica e atendimento dos interesses de múltiplos usuários com diferentes propósitos, as infra-estruturas de dados espaciais constituem um avanço.

2.2 Repositórios de Informação Geográfica

A partir dos recursos tecnológicos citados e do estabelecimento de acordos de cooperação, muitas agências nacionais de mapeamento criaram repositórios de informação geográfica. Esses repositórios são componentes voltados para a Internet que facilitam o acesso aos dados geoespaciais, concentrando-os em um único local, onde dados de diversas fontes podem ser encontrados. Esses repositórios também oferecem serviços para pesquisa, visualização e organização de dados espaciais [CBRW04], e dessa forma permi-

tem que provedores de dados tornem seus dados disponíveis e conhecidos aos usuários, através de descrições sobre os dados (*metadados*) e instruções sobre como acessá-los e usá-los.

De acordo com Crompvoets et al [CBRW04], há diferentes definições para o que é um repositório de informação geográfica. Mais recentemente, tais repositórios têm sido descritos de forma similar ao conceito de portais Web, e definidos como um local onde serviços são divulgados ou através do qual esses serviços são acessados [INS02]. A ênfase em serviços é recente se comparada a definições anteriores, baseadas na combinação entre ferramentas técnicas, mecanismos de cooperação interinstitucional e questões comerciais [FGD97].

O primeiro repositório de informação geográfica conhecido foi criado pelo Comitê Federal de Dados Geográficos (FGDC, do inglês *Federal Geographic Data Committee*), em 1994 nos EUA [DA05]. Esta iniciativa tinha o objetivo de principalmente reduzir esforços redundantes na coleta de dados. Mais tarde, no Brasil, a primeira fonte de dados espaciais publicamente disponíveis na Internet tornou-se operacional: o projeto GeoMinas[1] [DA05]. O GeoMinas consolidou um grande trabalho de descoberta, catalogação, preparo e disseminação de dados usados no estado de Minas Gerais [DA06]. Seus grupos temáticos discutiram a criação de um catálogo distribuído de metadados, um formato de intercâmbio de dados, o conteúdo de um mapa-base do estado, e pesquisas sobre meios tecnológicos adicionais para disseminação de informação, incluindo SIG baseados na Web, ainda na primeira metade da década de 1990.

Exemplos de repositórios de informação geográfica em outros países do mundo são o *National Geospatial Data Clearinghouse* (EUA)[2], GIgateway (Reino Unido)[3], *Nationaal Clearinghouse Geo-Informatie* (Holanda)[4], e o *Australian Spatial Data Directory* (Austrália)[5]. Entretanto, entre 67 repositórios analisados e comparados entre si por Crompvoets et al (2004) [CBRW04], foi percebida a existência de uma insatisfação crescente entre usuários desse tipo de serviço em relação às suas funcionalidades. Esse estudo então indicou que o foco deveria mudar de uma visão orientada a dados para uma visão orientada aos usuários e aplicações, algo que pode ser alcançado pelo uso de arquiteturas orientadas a serviços em infra-estruturas de dados espaciais [BC05, DA05].

[1]URL: http://www.geominas.mg.gov.br/
[2]URL: http://fgdc.ftw.nrcs.usda.gov/
[3]URL: http://www.gigateway.org.uk/
[4]URL: http://www.ncgi.nl/
[5]URL: http://asdd.ga.gov.au/

2.3 Geoportais

A palavra portal tem sido usada nos últimos anos com o significado geral de um "ponto de entrada" para informação e serviços disponíveis na Web, isto é, um Web site através do qual outros sites são encontrados. Portais são coleções organizadas ou referências a ítens de interesse dos usuários. Aplicando este conceito a geoinformação, um geoportal é, portanto, um "Web-site que apresenta-se como ponto de entrada para conteúdo geográfico na Web" [Tai05]. As funcionalidades de interesse dos geoportais incluem (1) descoberta de fontes de informação e contéudo, e (2) acesso on-line a aplicações.

Exemplos de geoportais existentes são o *Geospatial One-Stop* nos EUA, o *National Geospatial Data Framework* [BLM05] e o *MultiAgency Geographic Information for the Countryside* (MAGIC) [AEMS05], ambos do Reino Unido, e o EU-Geoportal, este como parte do projeto INSPIRE (*Infrastructure for Spatial Information in Europe*) [INS02].

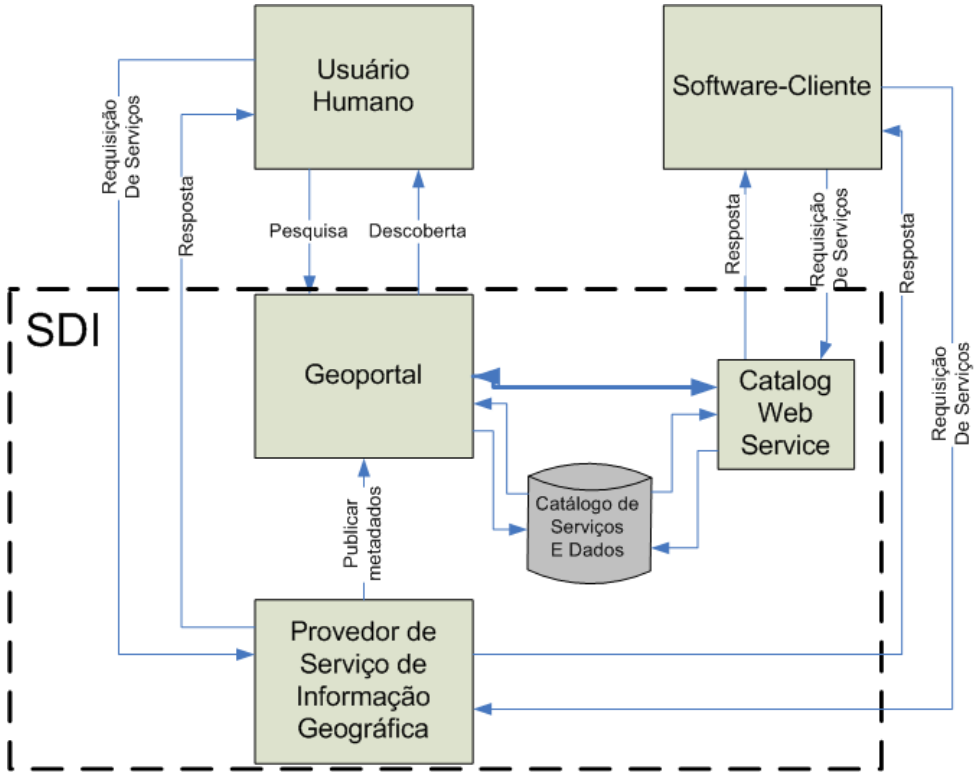

Figura 2.1: Geoportais como componentes de SDI [DA05]

Como geoportais são normalmente associados a uma ou mais infra-estruturas de dados espaciais, é importante estabelecer uma distinção de conceitos. Considera-se que uma SDI é formada pela confluência de diversos provedores de dados geográficos, cada qual garantindo acesso a dados através de serviços Web específicos [Wor04]. Para selecionar quais dados e, como conseqüência, quais serviços devem ser acessados para atender a uma

necessidade, o usuário busca por dados e serviços geográficos em um repositório. Naturalmente, os provedores de tais dados e serviços necessitam ter metadados correspondentes previamente incluídos neste repositório. No caso de um usuário humano, pesquisas são feitas interativamente através de um geoportal, possivelmente usando interfaces de pesquisa e outras ferramentas interativas; no caso de um software-cliente, isto pode ser feito através de um serviço Web de catálogo. Portanto, geoportais podem ser considerados componentes de uma SDI, o que é ilustrado na figura 2.1.

2.4 Infra-estruturas de Dados Espaciais

Infra-estruturas de Dados Espaciais (SDI, do inglês *Spatial Data Infrastructures*) constituem um conjunto de políticas, tecnologias e padrões que relacionam uma comunidade de usuários de geoinformação e atividades de suporte para produção e gerenciamento de informação geográfica [PWE99]. A idéia por trás das SDI envolve eliminar esforços redundantes e reduzir custos de produção através do compartilhamento de recursos, tanto relacionados a dados novos quanto a dados existentes. Para atender a esse propósito, é muito importante que vários parceiros, tendo interesses comuns, aceitem regras e sejam autorizados a fazer uso de dados ou informação produzidos por outros [CFRW01].

De acordo com Maguire e Longley [ML05], a expressão "infra-estrutura de dados espaciais" foi proposta pelo *Mapping Sciences Committee* do *National Research Council* dos EUA em 1993. Ela foi usada inicialmente para descrever o provimento de acesso padronizado a informação geográfica, mas muito do debate sobre o conceito reflete o conteúdo ideal de uma SDI nacional (ou *National SDI*, NSDI, que é também a sigla da *National Spatial Data Infrastructure* norte-americana, criada em 1994).

Muitas iniciativas de criação de repositórios de informação geográfica evoluíram para o que Masser [Mas99] chama de "primeira geração de infra-estruturas de dados espaciais", ao mesmo tempo em que observa que o uso do termo "infra-estrutura" indica a existência de alguma coordenação para formulação e implementação de políticas. Exemplos incluem a *Australian SDI* da Austrália[6], a *Canadian Geospatial Data Infrastructure* do Canadá[7], o *Sistema Nacional de Informação Geográfica* de Portugal[8] e a *National Spatial Data Infrastructure* dos EUA[9].

A primeira geração de SDI concentra-se em garantir um escopo temático amplo, o que está de acordo com os objetivos que permitem uma analogia entre SDI e outros tipos

[6]URL: http://www.ga.gov.au/nmd/asdi/
[7]URL: http://www.geoconnections.org/
[8]URL: http://snig.igeo.pt/
[9]URL: http://www.fgdc.gov/nsdi/nsdi.html

de infra-estrutura, particularmente o objetivo de financiar o desenvolvimento econômico através da garantia do acesso a produtos e serviços publicamente disponíveis e de múltiplos usos. "Publicamente disponíveis" não significa "suportado pelo governo", o que implica que serviços em uma SDI podem, indistintamente, ser providos pelo governo, pela iniciativa privada ou até mesmo por cidadãos, e portanto podem ser suportados através de taxas, por algum fundo governamental ou por outros tipos de arranjos econômicos. Apesar das SDI serem condutoras de desenvolvimento econômico, algumas das iniciativas analisadas por Masser [Mas99] não garantem acesso ao setor privado, ou o fazem pela cobrança de uma taxa de uso como meio de recuperar algum ou todos os custos de criação e disseminação de dados. Por outro lado, existe um requisito legal americano que determina que os dados de suas agências sejam gratuitos para o público.

Muitas lições foram aprendidas na primeira geração de SDI. Como Maguire e Longley [ML05] observam, nos EUA o foco foi predominantemente técnico, como resultado do controle centralizado do USGS (*United States Geological Survey*, Serviço Geológico dos EUA). Isto resultou na carência de atenção a aplicações potenciais, o que contribuiu para uma aceitação insatisfatória em setores privados e do governo.

Um passo adicional foi possível pela rápida evolução de sistemas de informação baseados na Web nos últimos anos. Nos EUA, a iniciativa NSDI passou por uma revisão em 2002, quando passou a ser chamada *Geospatial One-Stop* (GOS) e objetivou prover acesso abrangente a informação geográfica através de um portal Web[10]. Esta iniciativa inaugurou o atual conceito de geoportais [ML05, Tai05].

Guiando o processo de padronização de tecnologia e, conseqüentemente, definindo os elementos-chave para infra-estruturas de dados espaciais, diversos padrões foram propostos pela OGC, através de um *framework* chamado *OGC Reference Model* [Per03]. Este *framework* foi implementado com informações sobre estados, países e sobre regiões comuns a mais de um país [Dee06, Dem06, Sky06], sendo que casos de SDI urbanas são ainda escassos. Além do mais, muitos deles não estão em conformidade com os padrões da OGC [TAM03, KSR05, MRZW06, DF06].

Adicionalmente, SDI podem ser implementadas por meio de cadeias de serviços [Ala03] e componentes de software integrados [GGR05] os quais são encontrados através de geoportais [ML05]. A ênfase em serviços tem crescido desde a emergência de serviços Web e da adoção de arquiteturas orientadas a serviço (AOS) [HS05, LPD05]. A adoção de serviços Web para garantir acesso direto a dados é a mais importante distinção entre SDI de primeira e de segunda geração.

[10]URL: http://www.geodata.gov

2.4.1 Infra-estruturas de Dados Espaciais de Escopo Urbano

Considerando a existência de interoperabilidade entre diferentes SIG, a informação geográfica de um ou vários parceiros pode ser consolidada e assim formar um importante conjunto de recursos para a tomada de decisão no governo de uma região específica, como uma cidade, ou em níveis mais elevados de administração, como um estado ou país. Neste caso, SDI podem ser vistas como um conjunto de "peças de montar" [RWHJ00], e hierarquias de SDI são construídas nesse conjunto através do intercâmbio e consolidação da informação proveniente de diferentes organizações. Assim, é constituída uma rede de fontes de informação geográfica, cada qual com abrangência de cidades, estados, países, e por fim regiões maiores, chegando até o nível global, como pode ser visto na figura 2.2. Nessa hierarquia, as camadas inferiores provêem informação detalhada que auxilia na construção e consolidação dos níveis superiores [JSTW02, Man06].

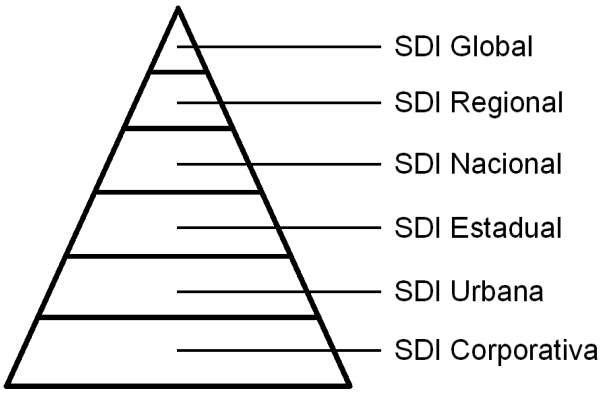

Figura 2.2: A visão "Blocos de Montar" das SDI [RWHJ00]

Denominamos as SDI que comportam detalhes de uma área urbana como *infra-estruturas de dados espaciais urbanas* (LSDI, do inglês *Local Spatial Data Infrastructures*), as quais estão na base de uma SDI global, como ilustrado na figura 2.2. Outra particularidade das SDI urbanas é o potencial de reunir um numeroso grupo de usuários, cada qual com necessidades distintas [DF06]. Esta característica é responsável por um aumento no nível de complexidade do desenvolvimento e distribuição de uma LSDI [BEK$^+$00, Mas05]. De fato, LSDI possuem valor para todos os outros níveis de infra-estrutura como fontes de informação detalhada [RWHJ00], e sua implementação deve pautar-se por demandas de novos grupos de usuários da informação geográfica, tais como turistas [KSR05], cidadãos [NBFRW04], pequenas organizações [TAM03] e outros [DF06, Mas05, MRZW06].

No entanto, o foco em SDI urbanas é recente, se comparado a estudos sobre SDI nacionais, regionais e globais [DA05]. SDI urbanas cresceram muito cedo e muito rapidamente em alguns países [BEK$^+$00], muitas vezes sob a forma de SIG urbanos, antes mesmo da

padronização de mecanismos de interoperabilidade. Assim, os esforços de padronização começaram nos níveis mais elevados de SDI, deixando LSDI heterogêneas e com dificuldades de acoplamento entre si e entre estas e as SDI de níveis superiores [JSTW02].

De fato, as SDI, em qualquer nível de detalhe, apresentam numerosas possibilidades de uso de serviços para encapsular dados de múltiplas fontes, e desta forma alcançar interoperabilidade real entre diferentes SIG. A importância de serviços para estas infra-estruturas inclusive levou Bernard e Craglia [BC05] a proporem uma nova tradução para a sigla SDI como *Service-Driven Infrastructures* (ou Infra-estruturas Dirigidas a Serviços).

2.5 Serviços Web

Todos os esforços anteriores de interoperabilidade ainda dependem de troca de dados *offline*, o que torna obrigatório o uso de réplicas distribuídas por todos os usuários da informação, exige maior espaço para armazenamento, pode apresentar dificuldade de controle de versões e de réplicas inconsistentes, além de causar um atraso entre a atualização da informação e sua disponibilização para os usuários.

Uma forma de minimizar estas dificuldades dá-se pela troca direta de dados entre sistemas através de uma rede de computadores, de modo que dados são requisitados diretamente de sua fonte, sem armazenamento temporário e em sua versão mais atualizada. Adicionalmente, o desenvolvimento de software baseado em componentes tem sido alvo de muito interesse devido ao seu potencial em reduzir custo e tempo de desenvolvimento, e devido ao interesse na implantação de sistemas distribuídos, os quais suportam a troca direta de dados [DA07]. De fato, um dos mais interessantes desenvolvimentos neste campo é a emergência de arquiteturas orientadas a serviço (SOA). SOA representam arquiteturas através das quais aplicações são projetadas pela composição de serviços, os quais são tarefas bem definidas, repetitivas e normalmente dirigidas a dados. As SOA, porém, mudam o foco em dados que atualmente existe em sistemas de informação para um foco em processos de alto nível fracamente acoplados.

A base das SOA é constituída por *serviços*, com suas descrições e operações fundamentais tais como publicação, pesquisa, seleção e invocação. Adicionalmente, SOA suportam aplicações maiores pela melhor distribuição de dados e de capacidade de processamento, pois facilitam a alocação de programas e outros recursos computacionais distribuídos em redes de computadores. Nesta arquitetura, serviços são auto-contidos, o que significa que informações sobre o serviço, incluindo funcionalidades, interfaces e comportamento, podem ser obtidas através de funções padronizadas oferecidas pelo próprio serviço.

Produtores, provedores, catálogos e consumidores de serviço são os atores, ou enti-

dades de negócio, que participam do modelo conceitual de serviços (Figura 2.3), onde a publicação é executada por um provedor de serviço, e consiste na criação de uma descrição do serviço em WSDL (o que inclui seus métodos, os quais se referem às funcionalidades do mesmo) e a publica nos portais de pesquisa na Web. Estes portais usam um serviço de registro para catalogar os serviços de terceiros juntamente com detalhes sobre os mesmos que permitem aos consumidores encontrá-los. Então, consumidores e produtores podem selecionar estes serviços utilizando-se do serviço de registro padronizado para procurá-los e também para obter o descritor de serviço em WSDL. Através da informação presente no descritor WSDL, um software-cliente pode ser criado para as operações de invocação. Neste ponto, o cliente pode realizar comunicação direta com o provedor de serviço através da Internet, usando os protocolos HTTP e SOAP e invocando os métodos do serviço que foram definidos no documento WSDL. As invocações terminam com a recepção da resposta esperada do serviço em uma linguagem de formatação, como é o caso do XML.

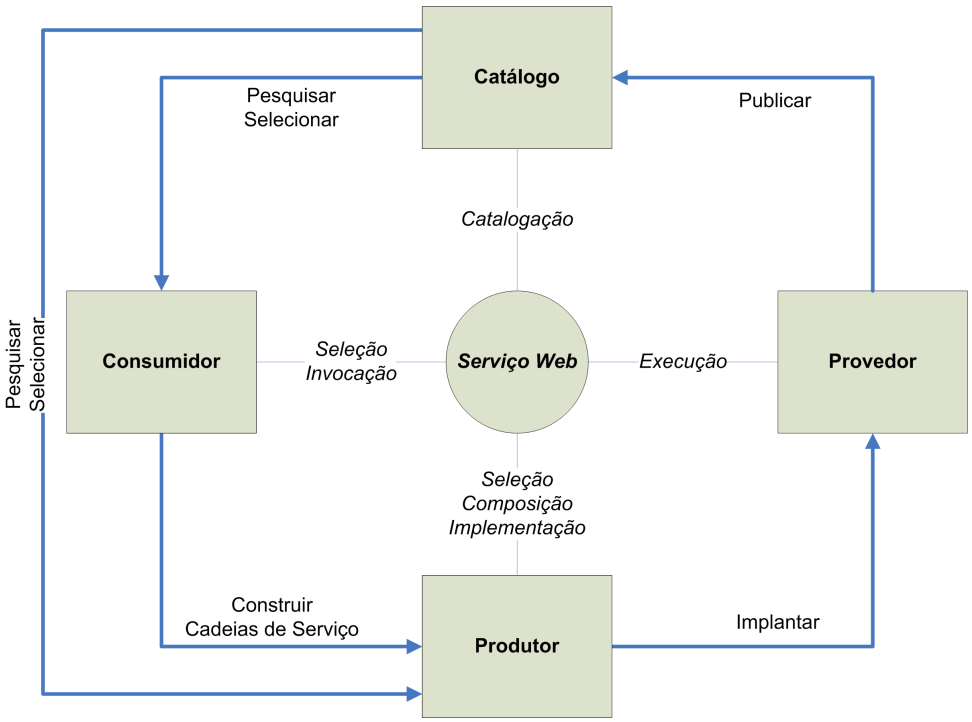

Figura 2.3: Modelo conceitual de serviços [DA07]

Consumidores de serviço podem criar software-clientes os quais acessam serviços através de uma rede de comunicação. Além disso, os consumidores podem tornar-se produtores quando agregam serviços de um ou mais provedores em uma cadeia, para oferecer resultados de um serviço composto para seus parceiros. Tais cadeias de serviço são transparentes para os consumidores que as usam.

Adicionalmente, serviços podem ser implementados independentemente de linguagens de programação ou ambientes de computação particulares de provedor, tecnologias essas que garantem a independência entre provedores (que executam os serviços) e consumidores (que invocam os serviços). Assim, serviços tornam-se uma alternativa interessante para alcançar interoperabilidade entre sistemas de informação diferentes.

Particularmente em aplicações geoespaciais, um mecanismo de interoperabilidade pode ser implementado a partir de padrões de codificação específicos para dados geográficos. Isto favorece a integração de diversas fontes de informação geográfica sem a necessidade dos procedimentos exportar-converter-importar, comuns no passado dos SIG.

O conceito de serviços Web é derivado do conceito de serviços em redes de computadores, no qual um serviço corresponde a uma função ou a uma interface entre duas camadas de abstração no projeto de redes [DA07]. Neste contexto, uma camada de baixo nível oferece serviço para uma camada de nível superior e a camada de nível superior não é afetada caso algumas propriedades das camadas inferiores sejam modificadas. A idéia de isolamento de detalhes técnicos específicos através do uso de diferentes níveis de abstração também se aplica no projeto de sistemas operacionais, sistemas de gerenciamento de banco de dados, projeto de software e em sistemas distribuídos.

As primeiras iniciativas de desenvolver sistemas distribuídos, na década de 1990, adotaram um conjunto limitado de tecnologias (protocolos de rede, sistemas operacionais e linguagens de programação), com o objetivo de reduzir dificuldades de implementação. Este foi o caso de implementações baseadas em chamadas remotas de procedimentos (*remote procedure calls*, RPC). Entretanto, era ainda necessário permitir que os sistemas baseados em linguagens de programação diferentes ou desenvolvidos para sistemas operacionais diferentes trocassem dados diretamente entre si [TA90]. Os próximos passos para garantir interoperabilidade entre sistemas heterogêneos envolveram esforços já citados anteriormente, como transferência de dados com conversão de formatos e definição de formatos específicos para intercâmbio em determinados domínios, o que inclui o domínio de geoinformação.

2.5.1 Padrões da Indústria para Serviços Web

Os serviços Web pertencem a uma classe específica que usa padrões abertos de Internet, tais como conexão e comunicação através de *Hypertext Transfer Protocol* (HTTP) e *Simple Object Access Protocol* (SOAP), identificação através de *Uniform Resource Identifier* (URI), formatação de conteúdo através de *Extensible Markup Language* (XML), e descrições dos serviços definidas através da *Web Services Definition Language* (WSDL).

A *World Wide Web Consortium* (W3C) é a organização que coordena o desenvolvimento de padrões populares tais como a *Standard Generalized Markup Language* (SGML) e a *Hypertext Markup Language* (HTML), lançadas respectivamente em 1986 e 1993, e que propôs em 1998 a linguagem XML como o padrão que poderia servir como principal formato de intercâmbio entre aplicações baseadas em Internet.

No campo dos sistemas de informação geográficos, a *Open Geospatial Consortium* (OGC) foi criada em 1994 e tem liderado iniciativas de interoperabilidade no domínio da geoinformação desde então. Entre as iniciativas da OGC está a *Geographic Markup Language* (GML), uma gramática XML para especificação e codificação de dados geográficos, publicada no ano 2000.

Em outra linha de desenvolvimento, alguns protocolos baseados em XML foram desenvolvidos com o objetivo de representar interfaces de métodos remotos (é o caso da WSDL, em 2002) e de representar invocações (é o caso do SOAP, em 2003). Tais protocolos tornaram-se a base tecnológica dos primeiros serviços Web.

2.5.2 Serviços Web Geoespaciais

Enquanto serviços Web genéricos provêem interoperabilidade entre sistemas de diferentes domínios, serviços Web geoespaciais introduzem propriedades geoespaciais a serviços e dão um passo a frente para facilitar a troca de dados e serviços geográficos entre instituições através da Internet, o que é garantido pela distribuição de recursos geográficos entre várias fontes de dados. Assim, serviços Web geoespaciais podem reduzir esforços redundantes que ocorrem quando dados espaciais são criados e disseminados, ao mesmo tempo em que contribuem para o reuso de código entre diferentes aplicações.

Basicamente, serviços Web geoespaciais diferem dos serviços Web genéricos pela presença de dados geográficos como entrada (por exemplo, um retângulo envolvente), como saída (por exemplo, um mapa-base de uma cidade), no processamento (por exemplo, verificar se duas ruas se cruzam), ou na combinação entre os anteriores.

A OGC propôs uma arquitetura para distribuição de dados e funcionalidades geográficos através da Internet, a qual abrange formatos de dados, métodos e especificação de interfaces para os diferentes serviços que a constituem. Esta arquitetura é chamada *OpenGIS Services Framework* [Per03]. A OGC também conduziu uma iniciativa para especificação de um conjunto de serviços específicos para ambientes urbanos chamado *Open Location Service* (OpenLS) [Mab04], no qual serviços tais como diretório (serviço de "páginas amarelas"), roteamento, geocodificação e outros são agrupados. Outros serviços de alto-nível para ambientes urbanos foram propostos em [DA05] e a variedade de pos-

sibilidades de usar tais serviços urbanos na construção de aplicações úteis e compactas é extensa.

No entanto, o *OpenGIS Services Framework* não usa necessariamente os padrões de serviços Web mais comuns, tais como SOAP e WSDL. A adoção destes padrões é inclusive desejável para garantir interoperabilidade entre serviços Web genéricos e serviços Web da OGC. Ao invés de usar *Universal Description, Discovery and Integration* (UDDI), a OGC propõe o uso de serviços de catálogo para as operações de publicação, pesquisa e seleção [Son04]. Os serviços Web da OGC também têm interfaces específicas para invocação conforme o objetivo de alto-nível do serviço, sendo que as interfaces não são anotadas em descritores (em WSDL, por exemplo) por terem a assinatura padronizada. Esta alternativa cria dificuldades para a indexação e procura por serviços em catálogos não OGC. Além disso, alguns serviços da OGC usam GML para codificar e distribuir dados, enquanto serviços Web comuns usam XML genérico, o que na verdade não é uma grande diferença, uma vez que GML é derivada da XML [Per03]. Definitivamente, as principais diferenças existentes entre serviços da W3C e da OGC devem ser sanadas tão logo seja possível, pois isto incentivaria a adoção dos padrões propostos por ambas as organizações de uma forma mais ampla [Son04].

Alguns serviços fundamentais foram especificados pela OGC, como serviços para registro, composição, visualização e codificação de informação geoespacial. Estes podem então ser combinados para formar sistemas de informação distribuídos com funções geográficas importantes. Os principais serviços são descritos em seguida.

Web Feature Service: provê uma interface para inserir, selecionar, atualizar e remover feições geográficas (objetos).

Web Coverage Service: provê acesso a geocampos de forma semelhante ao Web Feature Service. Porém, sua resposta não representa imagens de geocampos, mas detalhes sobre os mesmos.

Web Gazetteer Service: estende o *Web Feature Service* com recursos para a implementação de interfaces para *gazetteers* [SDB+05].

Web Registry Service e **OpenGIS Catalog Service (OCS)**: implementam um serviço com funcionalidade similar ao UDDI para serviços Web.

Web Coordinate Transformation Service: provê um algoritmo que converte coordenadas de objetos geográficos entre diferentes sistemas de referência geoespacial.

Web Map Service: produz mapas para a Web. Mapas, neste serviço, são imagens e não apresentam os dados geográficos, mas apenas uma representação pronta.

Web Terrain Service: similar ao *Web Map Service*, este serviço cria representações em três dimensões de superfícies. Tanto este serviço quanto o *Web Map Service* produzem

representações em formatos de imagem ou em formatos vetoriais tais como o formato *Scalable Vector Graphics* (SVG).

Interfaces de serviços Web tem sido desenvolvidas pela OGC até mesmo para redes de sensores, sob os nomes **Sensor Collection Service** e **Sensor Planning Service** [Ope]. Estes serviços, além de outros, consolidam uma arquitetura aberta e flexível, a qual encontra aplicação em situações muito similares àquelas discutidas para SDI. Além dos argumentos citados anteriormente para SDI, as infra-estruturas de dados espaciais devem ainda ser distribuídas, suportar múltiplas aplicações e múltiplos clientes de diferentes tipos, ser compostas por múltiplas fontes de dados, ser gerenciadas por mais de um grupo para manutenção dos dados, conseqüentemente em um ambiente computacional heterogêneo. Adicionalmente, não é adequado que uma SDI imponha a adoção de produtos específicos aos seus participantes mas, ao contrário, que apenas determine um conjunto mínimo de padrões a serem seguidos. Estes padrões precisam ser largamente aceitos, e os padrões típicos de Internet, como os que foram mencionados nesta seção, atendem adequadamente este quesito.

No entanto, mesmo se os serviços Web geoespaciais adotassem integralmente os padrões da OGC e da W3C, as dificuldades técnicas de implementação de SDI e SIG distribuídos não necessariamente seriam reduzidas. Um exemplo disso é a necessidade que SIG têm de requisitar informações de processamento mais demorado, em sua maioria, o que apresenta peculiaridades se comparados a outros tipos de sistemas de informação. Neste caso, como os serviços da W3C [Wor04] usam o protocolo HTTP em sua camada de aplicação, e este protocolo suporta apenas chamadas síncronas, este fato representa um problema quando transações longas existem.

Algumas iniciativas tentaram emular comunicação assíncrona em serviços Web através do uso de *listeners* que recebem respostas e as encaminham para o cliente que a requisitou [RLT05], normalmente sobre protocolos diferentes do HTTP [BCPR04], como o protocolo de correio eletrônico SMTP [CPD06]. Todavia, até onde foi pesquisado, nenhum padrão permite comunicação assíncrona usando apenas os padrões de serviços Web.

Com relação a informação geográfica especificamente, as especificações de serviços Web da OGC [Whi05] descrevem funções e componentes, dentre os quais comunicação assíncrona é suportada apenas por serviços de sensores [Ope] como o *Web Notification Service*, mas nenhum protocolo de comunicação foi definido nesta especificação.

Além da necessidade do exemplo, outros requisitos são essenciais para o desenvolvimento de SDI e SIG interoperáveis através de serviços Web. A adequação dos serviços Web para o atendimento desses requisitos é avaliada no próximo capítulo.

Capítulo 3

Metodologia e Infra-estrutura de Avaliação

O objetivo desta pesquisa é investigar a adequação dos padrões OGC para a implementação de sistemas de informação geográficos baseados em dados compartilhados em rede, infra-estruturas de dados espaciais de âmbito local e softwares-cliente para tais SIG e LSDI, assim como definido na seção 1.3.

Para a consecução da investigação, um cenário de uso foi especificado na seção 3.1. Em seguida, na seção 3.2 foi especificado um protótipo de infra-estrutura de dados espaciais e de SIG urbano, onde também são apresentadas duas situações de teste, a primeira com a intermediação por um provedor de serviços geográficos incompatível com o OGC e a segunda implementação com o uso direto de serviços geográficos compatíveis com o OGC por parte do cliente. Na seção 3.3 foram detalhados os serviços geoespaciais implementados e o banco de dados geográfico adotado. Finalmente, são apresentadas as limitações identificadas na seção 3.4, a partir das duas situações no cenário ilustrado.

3.1 Cenário de Implementação e Estudo de Caso

O Modelo de Referência do OGC [Per03] define um cenário de uso intitulado "assistência a viagem do cliente" (*consumer travel assistance*) [p. 83] através do qual um consumidor de informação geográfica usa um dispositivo móvel, onde há um cliente de SIG, para (1) obter sua posição atual ou localização; (2) obter um endereço de destino, dado um número de telefone; (3) obter uma localização, dado o endereço de destino; (4) obter uma rota entre a origem e o destino; (5) determinar as condições de tráfego, clima e condições da rodovia ao longo do caminho; e finalmente (6) obter avisos em tempo real sobre as condições da viagem. O cenário de uso serve como gabarito para especificação e

testes de sistemas distribuídos [Pre05].

Esta tarefa requer um conjunto significativo de dados geográficos, os quais possivelmente pertencem a diferentes provedores de informação. Um modelo computacional possível é garantido pela presença de provedores deste tipo de serviço que intermedeiam todas as transações entre o cliente e os serviços, enquanto também é possível que o cliente ocupe-se das requisições e recepção de respostas dos diversos serviços que compõem a tarefa. Em ambos os casos há impactos diretos no desempenho do cliente e na facilidade com a qual softwares-cliente são desenvolvidos para um SIG qualquer.

Para nossa prova de conceito, detalhamos e estendemos o cenário "assistência a viagem do cliente", porém retiramos os passos 5 e 6 do cenário original. Essas alterações no cenário de uso tiveram o objetivo de deslocar a complexidade de serviços de alto nível não especificados até o momento pelo OGC para aqueles serviços que já possuem especificação aprovada. Dessa maneira, foi reduzida a influência de decisões de projeto e implementação desta pesquisa sobre aqueles serviços para os quais o OGC não emitiu especificações de implementação. O cenário do ORM alterado possui os passos enumerados abaixo e é ilustrado na Figura 3.1.

1. Obter a posição de origem ou localização do cliente ao fornecer um dos seguintes dados: um telefone fixo próximo, um cruzamento entre pelo menos dois logradouros, ou uma referência baseada em nome ou local conhecido (Informação A na Figura 3.1).

2. Obter um endereço de destino, dado um número de telefone.

3. Obter a localização de destino, dado o endereço de destino (Informação B na Figura 3.1).

4. Obter uma lista de pontos de interesse próximos à localização de destino (Informação C na Figura 3.1, representada pelo conjunto de pontos no interior do círculo).

5. Selecionar um ponto de interesse (Informação D na Figura 3.1).

6. Obter uma rota entre a posição de origem e a posição de destino, passando obrigatoriamente pelo ponto de interesse (Informação E na Figura 3.1, representada por uma linha que parte do ponto de origem e segue até o ponto de destino).

A implementação de dois modelos de computação é descrita nas próxima seção e as limitações identificadas a partir desta implementação são discutidas na seção 3.4.

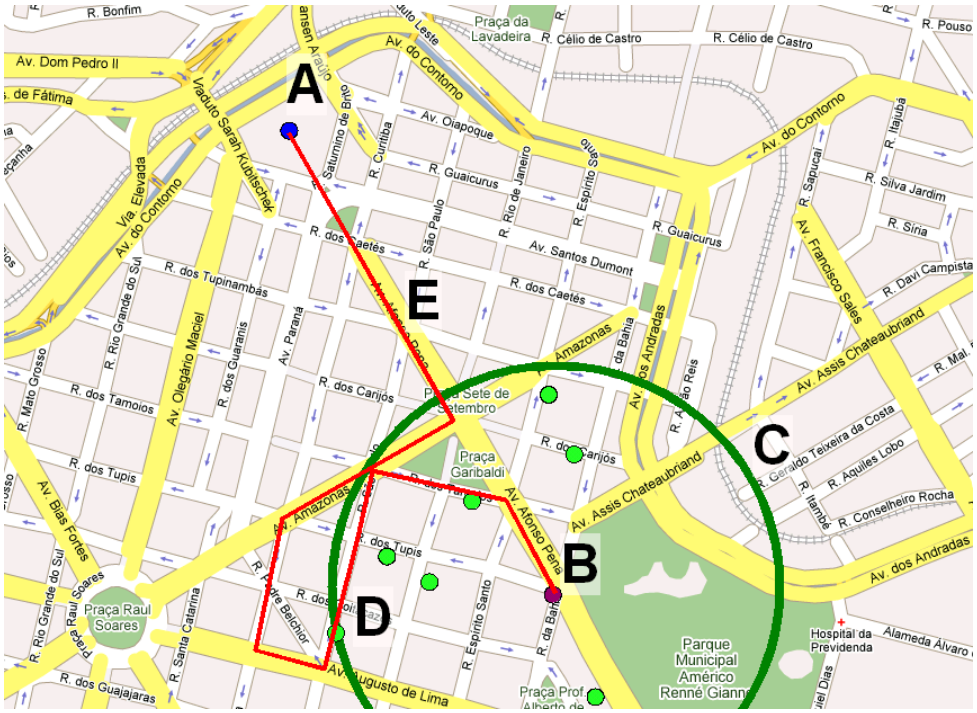

Figura 3.1: Amostra de resultado do processamento de uma consulta

3.2 Infra-estrutura de Dados Espaciais Urbanos

3.2.1 Serviços de Alto Nível

O exemplo de infra-estrutura de dados espaciais urbanos utiliza-se de informações geográficas disponíveis sobre a cidade de Belo Horizonte (MG). Apesar de as informações terem sido usadas na forma como estão disponíveis na cidade, a LSDI já existente não é compatível com os serviços do OGC e baseia-se em um acordo de cooperação entre diferentes organizações públicas, privadas e na troca *offline* de arquivos centralizados em um repositório mantido pelas organizações participantes do convênio [DF06].

A LSDI de testes é composta por alguns dos serviços de nível conceitual propostos originalmente no estudo de Davis Jr e Alves [DA05], os quais são discriminados abaixo.

- Dois serviços do tipo *Mapa-base*, servindo o segundo para simples replicação;
- Um serviço do tipo *Roteamento*;
- Dois serviços do tipo *Geocodificação*, servindo o segundo como substituto não equivalente ao primeiro;
- Um serviço do tipo *Diretório* para números de telefone;

- Um serviço do tipo *Diretório* para empresas;
- Um serviço do tipo *Localização Pessoal*;
- Um serviço personalizado para intermediar a comunicação entre os serviços anteriores e os softwares-cliente;
- Um serviço de *Catálogo*.

Cada um deles foi implementado como um serviço independente dos outros, mas em alguns casos compartilhando o ambiente do mesmo servidor HTTP, como ilustrado na Figura 3.2. O serviço personalizado foi desenvolvido para representar o papel de um *integrador de serviços* em cadeias, sobre as quais o software-cliente não possui qualquer controle.

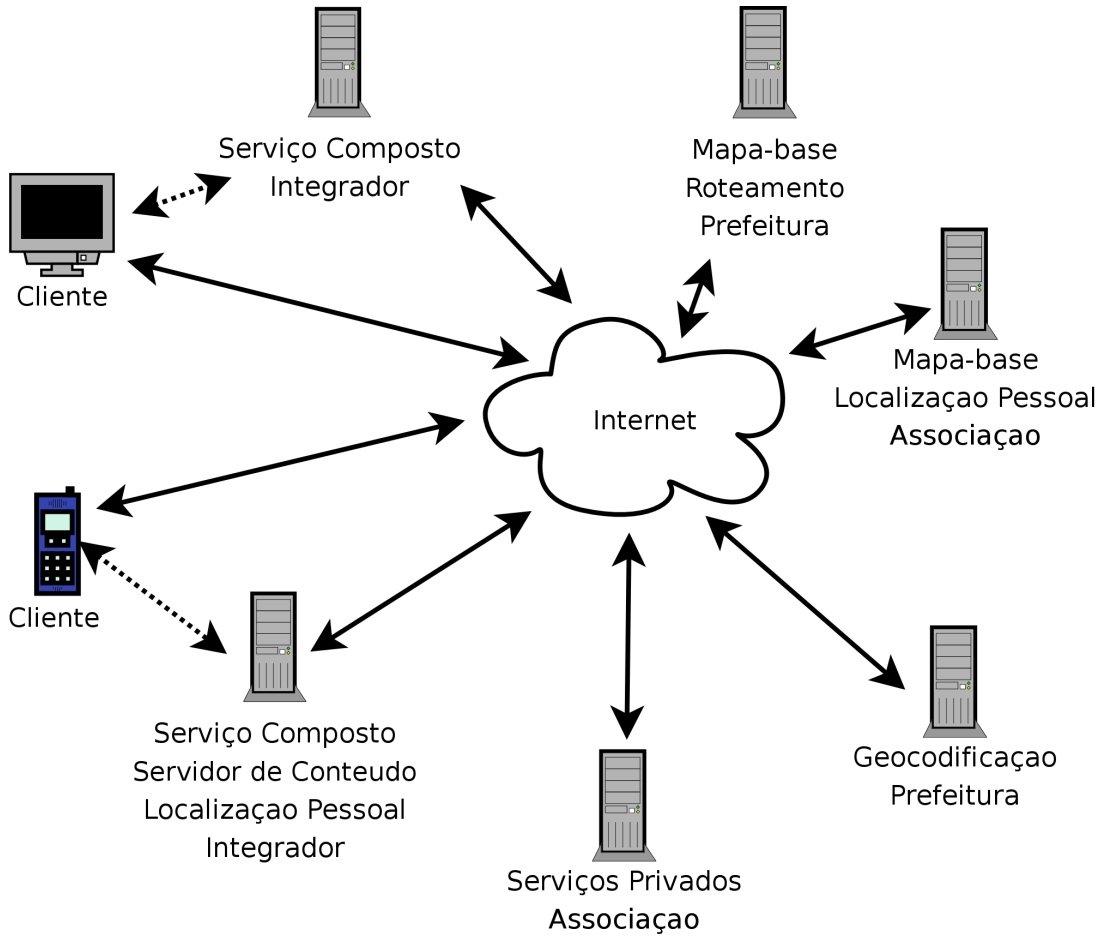

Figura 3.2: Distribuição física dos serviços implementados

Os serviços de nível conceitual foram implementados sobre serviços propostos pelo

OGC em suas especificações *OGC Web Services* e *Open Location Services*, o que será tratado na seção 3.3.

3.2.2 Implementação 1

Através da primeira implementação é avaliada a facilidade com a qual são desenvolvidos serviços Web regulares, no padrão W3C, derivados dos serviços Web geoespaciais propostos pelo OGC, ao mesmo tempo em que se avalia o desenvolvimento de clientes que usam serviços genéricos ao invés de serviços padrão OGC. Nesse caso, o cliente usa o serviço Web derivado, e cada novo serviço desse tipo exige alterações no cliente que o utiliza.

O conjunto de funções do serviço derivado é exposto para o cliente através de sua respectiva interface, e funciona como uma API para os serviços primitivos, próprios ou de terceiros, adotados pelo provedor.

A Figura 3.3 mostra um servidor denominado Integrador, onde localiza-se o serviço derivado, e a segunda camada de servidores, onde localizam-se as fontes originais de informação e funções geográficas.

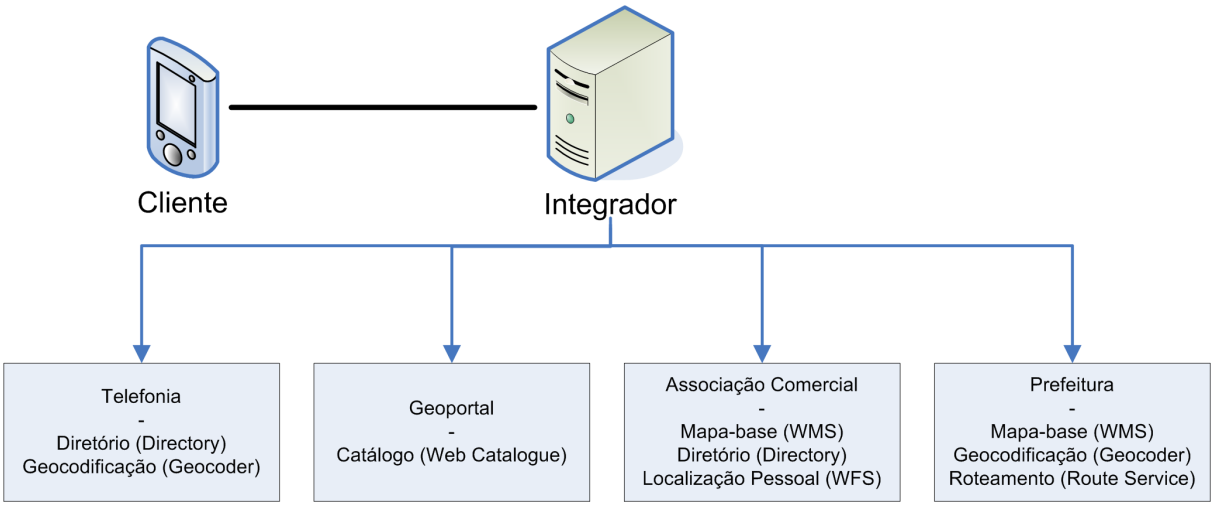

Figura 3.3: Implementação 1, onde há a intermediação de um provedor

Ao implementar esse modelo, o cliente não conhece a existência da segunda camada de servidores e serviços e não possui controle sobre os mesmos.

3.2.3 Implementação 2

Diferentemente da primeira implementação, na segunda não são construídos serviços Web derivados, ou seja, somente serviços Web geoespaciais são disponibilizados por seus respectivos provedores, os quais são acessados diretamente pelos softwares-cliente. Assim, uma vez que esses serviços são padronizados, o suporte a novas funções padronizadas nos serviços não provoca alterações no software-cliente. Por esse motivo, através desta implementação foi avaliada principalmente a facilidade com a qual são desenvolvidos softwares-cliente para serviços primitivos. A Figura 3.4 mostra a camada onde localizam-se as fontes originais de informação e funções geográficas.

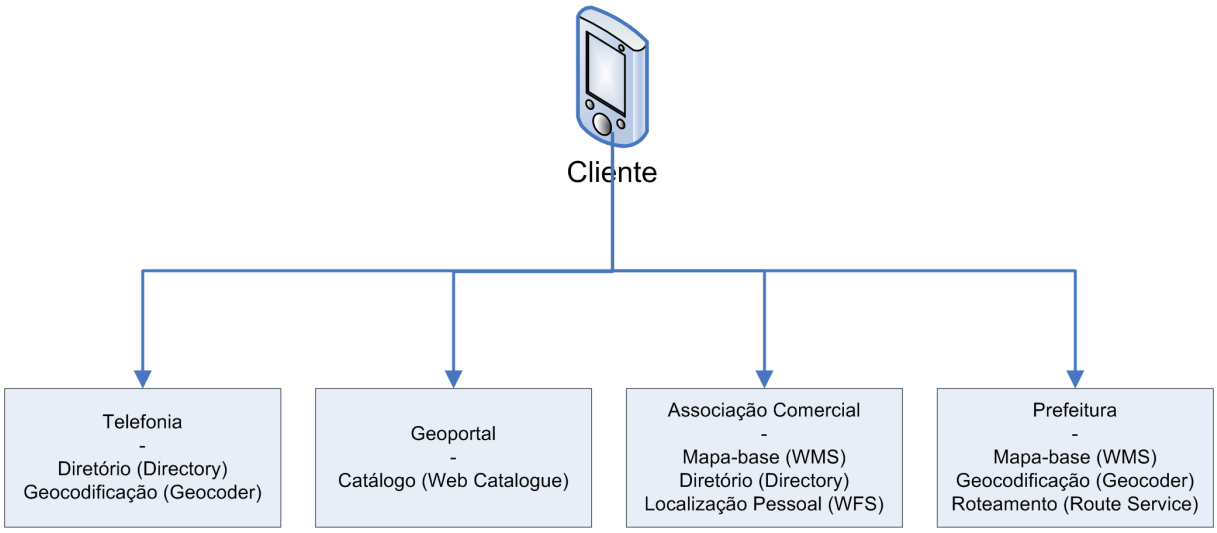

Figura 3.4: Implementação 2, onde o cliente acessa diretamente os serviços alvo

O que diferencia a primeira da segunda implementação é principalmente o fato de que o segundo cliente usa interfaces comuns a todos os serviços compatíveis com o OWS e o OpenLS, isto é, temos um cliente universal. No primeiro caso, temos um cliente fortemente dependente de um provedor de serviços, ou seja, trata-se de um cliente específico para uma aplicação.

3.2.4 Clientes para o Sistema de Informação Geográfico

Os softwares-cliente são responsáveis pelo uso de serviços primitivos e ou derivados (compostos). Entretanto, os softwares-cliente reais são destinados a dispositivos com diferentes capacidades de processamento, armazenamento e comunicação, sendo classificados na literatura como magros, ricos ou gordos, em função da parcela de funcionalidades mantidas no cliente e no servidor [Pre05]. As definições clientes magro e cliente gordo também

são dadas aos próprios dispositivos, com relação às suas capacidades de processamento, armazenamento e comunicação.

Os dispositivos com menor capacidade de processamento são classificados como *clientes magros*. São normalmente alimentados por uma bateria e têm capacidade de energia limitada. Ao mesmo tempo, estes dispositivos também têm limitações de memória principal e memória secundária. Um exemplo são os telefones celulares modernos com uma máquina virtual Java específica (KVM). Esses equipamentos normalmente demandam serviços remotos através de um meio de comunicação de alto custo, se comparado a redes de comunicação corporativas e domésticas.

Como *clientes ricos* são classificados dispositivos nos quais há um processador mais rápido e a capacidade de memória principal e memória secundária pode ser estendida. Esses clientes normalmente possuem recursos computacionais suficientes para manter parte da aplicação no próprio dispositivo, enquanto outra parte do processamento é realizada remotamente. Os meios de comunicação normalmente usados são os mesmos de clientes magros, mas em alguns casos redes corporativas e domésticas são usadas. Exemplos de dispositivos assim são os *Personal Digital Assistants* e alguns telefones celulares específicos.

Por fim, como *clientes gordos* são classificados dispositivos com capacidade de processamento, armazenamento e de energia superiores ao necessário por aplicações comuns. Normalmente são computadores pessoais, com alimentação contínua de energia e equipados com disco rígido. Assim, mesmo computadores pessoais em movimento (*notebooks* alimentados por bateria e utilizando-se de uma rede sem fio) são clientes gordos, uma vez que a velocidade de processamento é mais relevante nesta classificação.

Nos três casos, o projeto de um software-cliente para um SIG deve ser dimensionado em função da capacidade do dispositivo que o executará. Para os experimentos, foi desenvolvido um único software-cliente que implementa as funcionalidades como *cliente magro* mas não possui restrições no processamento, memória, armazenamento e capacidade de comunicação de dados, pois os requisitos do modelo abstrato não são relacionados diretamente a essas variáveis.

3.3 Serviços Implementados

3.3.1 Serviços OGC

As especificações OWS e OpenLS atendem aos requisitos do Modelo de Referência do OGC, o qual divide-se em dois níveis de abstração: modelo abstrato, onde serviços

possuem requisitos e um comportamento genérico não dependente das tecnologias atuais; e modelo de implementação, onde é especificado como os serviços são desenvolvidos através das tecnologias de software atualmente disponíveis.

Os serviços OGC implementados foram *Web Feature Service*, *OpenGIS Catalog Service*, *Web Map Service*, *Geocoder Service* e *Route Service*, sendo os dois últimos do conjunto OpenLS [Ope05b, Ope05a, Mab04]. As funções destes e de outros serviços do OGC já foram apresentadas na seção 2.5.2.

Os serviços foram codificados em linguagem Java e o *framework* Apache Axis foi usado para dar suporte às mensagens SOAP como serviços Web, os quais são executados em servidores HTTP Apache Tomcat.

3.3.2 Serviços Não-OGC

Para o desenvolvimento de um protótipo que atendesse ao cenário de uso da seção 3.1 e a implementação dos serviços de alto nível especificados na seção 3.2.1, um conjunto de funções não compreendidas pelos serviços do OGC foi projetado e implementado.

O agrupamento de funções em serviços foi baseada no propósito de cada uma, ao mesmo tempo em que cada grupo deveria ser reunido em um único componente ou conjunto de componentes de software, o que favoreceria a implantação das funções.

O primeiro grupo foi denominado *Data Exchange Service* (DXS) e foi implementado como um serviço Web capaz de servir como repositório de informação geoespacial trocada entre os serviços OGC primitivos e os clientes. O objetivo do grupo é principalmente provêr persistência de dados e conseqüentemente permitir a recuperação de informação em uma segunda chamada, a qual é isolada da invocação onde é realizada a requisição ao serviço geoespacial. O serviço será detalhado na seção 4.1.

O segundo grupo foi denominado *Client Access Service* (CAS) e torna o cliente apto a responder por requisições de outros clientes e de serviços, como se o cliente fosse um servidor. O grupo foi projetado como um comportamento similar ao de servidores HTTP, o que torna possível que invocações sejam realizadas e recebidas pelo cliente, da mesma forma. O comportamento será detalhado na seção 4.2.

O terceiro grupo foi denominado *Transaction Control Service* (TCS) e foi implementado como um compilador XSLT que converte uma especificação de alto nível em XML para um código específico de uma LSDI, onde são introduzidos os serviços geoespaciais necessários. Finalmente, o código específico pode ser traduzido para código executável, em uma linguagem de programação alvo, no qual já são introduzidas particularidades do software-cliente e do dispositivo-cliente, como por exemplo a necessidade de uso do DXS.

O serviço será explicado na seção 4.3.

Finalmente, todos os serviços, compatíveis ou não com o OGC, compartilharam o mesmo esquema conceitual de banco de dados, o qual é especificado na próxima seção.

3.3.3 Banco de Dados Geográfico

Os dados geográficos usados pelos serviços Web foram armazenados em um servidor de banco de dados baseado no SGBD PostgreSQL com a extensão espacial PostGIS. Através dessa extensão espacial, algumas operações espaciais foram omitidas dos serviços Web e mantidas sob responsabilidade do SGBD, as quais são:

- `asgml`, para saída de dados geográficos no formato da GML
- `distance`, para cálculo de distância entre coordenadas geográficas
- `geomunion`, para consolidação ou agrupamento de objetos geográficos
- `intersection`, para obtenção de região resultado da interseção de dois polígonos
- `point`, para saída de dados geográficos no format latitude/longitude
- `setsrid`, para atribuição de identificação de sistema de referência geoespacial aos polígonos obtidos de diferentes provedores
- `touches`, para verificação de cruzamento de ruas
- `transform`, para transformação de sistema de referência geoespacial

Além disso, o SGBD foi compartilhado entre todos os serviços Web implementados, o que ampliou nosso foco na troca de mensagens e na implementação das funcionalidades específicas dos diferentes tipos de serviço OGC.

A figura 3.5 apresenta as entidades presentes no SGBD por meio do modelo OMT-G [CCD+05]: nomes de logradouro, trechos de logradouro, estabelecimentos comerciais, bairros, regionais, endereços e pontos de referência.

3.4 Limitações Identificadas

O documento do Modelo de Referência do OGC [Per03] define cinco perspectivas para o projeto de SIG distribuídos, as perspectivas *Organizacional*, *Informacional*, *Computacional*, de *Engenharia* e *Tecnológica*, cada qual abrangendo um conjunto de propriedades

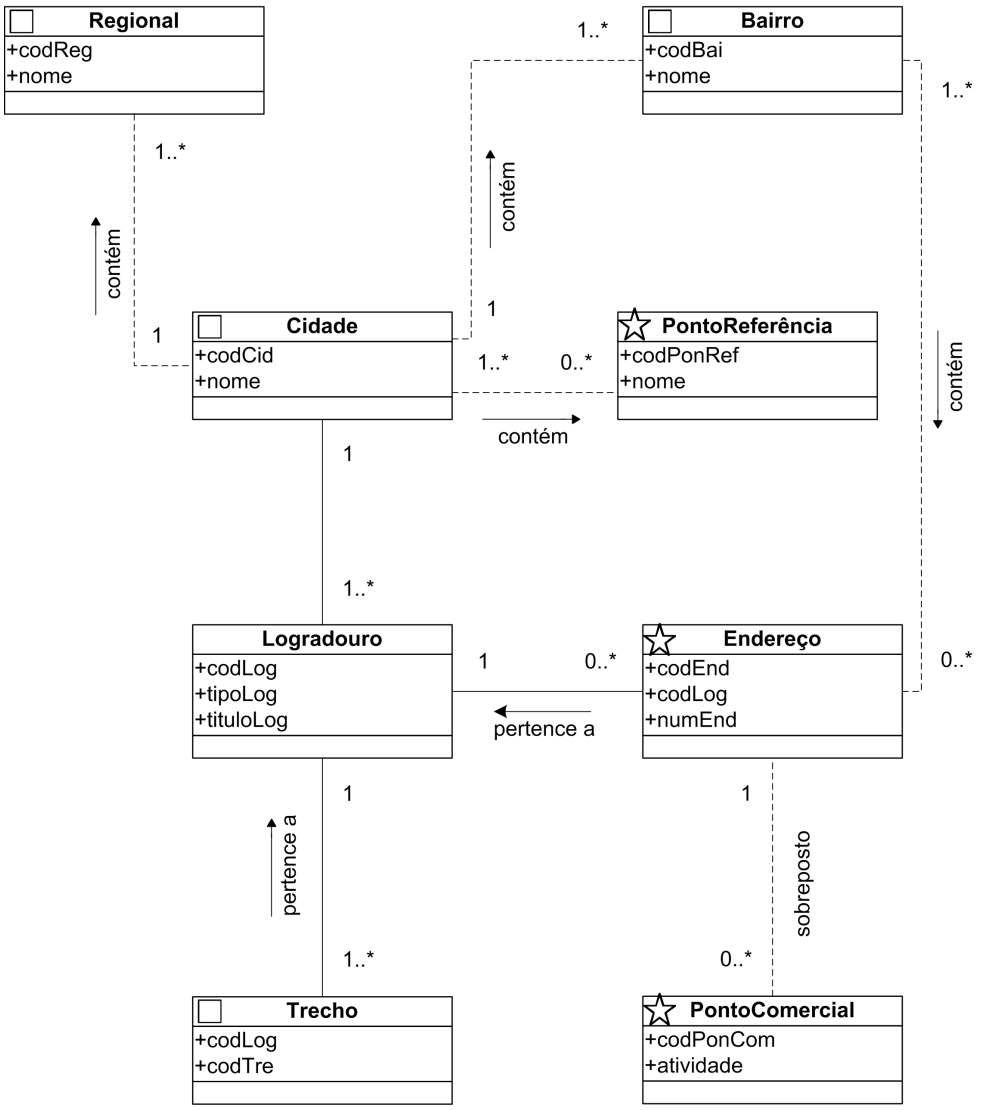

Figura 3.5: Modelo OMT-G do banco de dados geográficos usado

e requisitos do sistema. A perspectiva *Organizacional* descreve um sistema segundo seus propósitos dentro de uma organização, ou seja, a lógica de negócio representa uma de suas descrições mais importantes. Em seguida, a perspectiva *Informacional* descreve um sistema conforme seu conteúdo, tipos de informação, propriedades dos dados, principais fluxos entre unidades organizacionais, além de outros aspectos importantes. A perspectiva *Computacional*, por sua vez, descreve o sistema através de suas funções, operações, rotinas e transformações sobre os dados e informações. Somente a perspectiva de *Engenharia* assume a natureza distribuída do sistema e relaciona os objetos das perspectivas anteriores a componentes de software distribuídos em uma rede de computadores. Porém, tais componentes de software ainda são abstratos, independentes de tecnologia e apenas

são materializados em código executável através da próxima perspectiva. Finalmente, a perspectiva *Tecnológica* define como os componentes de software são implementados, implantados e como funcionam em um conjunto de computadores, onde, no contexto dos SIG distribuídos, a tecnologia de serviços Web é a mais comum.

Dentre outros objetivos, a perspectiva de *Engenharia* oculta aspectos complexos de parte do software, onde tais aspectos mostrem-se irrelevantes ao seu propósito. É frequentemente o caso de recursos de segurança, recuperação em caso de falhas, controle de privacidade, otimização de custos de comunicação, implementação de especificidades de diferentes tipos de dispositivos-cliente, e outros.

Através do desenvolvimento de clientes de SIG para os serviços e para o cenário especificado nas seções anteriores, foram identificadas limitações de projeto e de implementação das *transparências* definidas pelo ORM na perspectiva de *Engenharia*, que são:

- Transparência de acesso, que oculta diferenças na representação de dados e protocolos de acesso;
- Transparência de localização, que oculta a localização física de um componente de software;
- Transparência de migração, que oculta de um componente de software detalhes de relocação dele próprio;
- Transparência de relocação, que oculta das interfaces detalhes sobre relocação de outras interfaces usadas pelas primeiras;
- Transparência de replicação, a qual ocupa-se de detalhes sobre a criação de réplicas consistentes para aumentar o desempenho e a disponibilidade do sistema;
- Transparência de persistência, a qual ocupa-se da criação e uso de objetos no lado do servidor;
- Transparência de falha, que oculta falhas e recuperações do sistema diante dessas falhas;
- Transparência de transação, que oculta detalhes sobre a coordenação de tarefas entre um grupo de componentes de software.

Algumas das transparências, como transparência de acesso, são providas por serviços de infra-estrutura tais como o *domain name system* (DNS). Outras porém, mostram-se mais difíceis de implementar e suportar, tais como as transparências de migração ou relocação [Per03].

São discutidas nesta seção as limitações identificadas através da implementação 1 (I1), seção 3.2.2, e da implementação 2 (I2), seção 3.2.3. Estas limitações referem-se às transparências citadas anteriormente e foram analisadas sobre os requisitos não-funcionais (RNF) da perspectiva de *Engenharia* definidos no ORM [Per03]. Para a análise foi utilizada a técnica proposta em [MCN92], através da qual tornaram-se explícitos os RNF e as técnicas de satisfação, que são os mecanismos de suporte a estes RNF. Porém, uma técnica de satisfação que suporta um RNF pode prejudicar outro. Portanto, pelo resultado dos impactos positivos e negativos de determinadas técnicas de satisfação em RNF diversos, se deve renegociar requisitos e prioridades com os participantes ou usuários de um sistema, com o objetivo de torná-lo adequado para todos.

Após o nome de cada técnica de satisfação pode haver um elemento composto por um único par de colchetes. A inclusão de um nome de entidade entre colchetes (Ex: [cliente], [WFS], etc) indica o componente de software ao qual pertence o RNF afetado, sendo que [cliente] é a entidade do RNF caso o nome da entidade seja omitido.

Nas próximas seções são discutidos estes RNF de cada transparência desejada para o projeto de uma SDI. Inicialmente, na seção 3.4.1, são apresentados os RNF identificados nas especificações do OGC e avaliados neste trabalho. Na seção 3.4.2 é discutida a transparência de acesso, pela qual são usados protocolos de acesso a dados e serviços remotos. Em seguida, na seção 3.4.3 é discutida a transparência de localização, a qual é suportada principalmente pelo serviço de rede *domain name system*. Na seção 3.4.4 é discutida a transparência de migração que trata da relocação de componentes de software sem associar complexidade a implementação destes componentes. Em seguida, na seção 3.4.5 é apresentada a transparência de relocação, a qual oculta a complexidade do processo de acessar componentes que sofrem migração. Na seção 3.4.6 é discutida a transparência de replicação, pela qual réplicas são produzidas para aumentar a disponibilidade e desempenho do sistema sem aumento da complexidade. A transparência de persistência é apresentada na seção 3.4.7, onde o armazenamento e manipulação de objetos remotos foram avaliados. A transparência de falha é apresentada na seção 3.4.8, onde mecanismos de recuperação e suporte a disponibilidade em caso de falhas de software e hardware foram experimentados. E finalmente, na seção 3.4.9 é discutida a transparência de transação, pela qual são construídos mecanismos de coordenação de tarefas entre componentes de software distribuídos. Na seção 3.4.10, é realizada e discutida uma avaliação sobre o impacto dos mecanismos projetados, para todas as transparências anteriores, no processo de desenvolvimento de softwares-cliente para estas infra-estruturas.

3.4.1 Requisitos Não-Funcionais

Para cada componente de software especificado apresentam-se requisitos não-funcionais (RNF), os quais podem ser definidos como fatores implícitos de qualidade de software que podem ser medidos de forma indireta [Pre05]. Os RNF avaliados para os serviços prototipados são descritos abaixo.

Auditabilidade de componentes refere-se à capacidade de o cliente verificar as propriedades de componentes de software e de informação internos a uma cadeia de serviços, o que atualmente só é possível em um modelo de computação totalmente horizontal, sem uso de serviços compostos, o que é similar ao modelo da implementação 2 definido na seção 3.2.3. Através desse modelo, o cliente tem acesso direto aos serviços Web e tem como controlar diretamente parâmetros tais como desempenho e qualidade de informação. A medida deste RNF se dá pela disponibilidade ou indisponibilidade dos metadados sobre os componentes usados por serviços Web geoespaciais invocados.

Controle sobre componentes refere-se à capacidade de obtenção de informação sobre componentes serviços a partir do catálogo. A medida deste RNF se dá pela possibilidade de realizar a recuperação de informação sobre serviço no catálogo a partir da URI do serviço.

Economia de comunicação refere-se ao número de invocações realizadas pelo cliente para obter resultado de um serviço Web geoespacial. O número adequado de invocações é dependente da técnica, da transparência associada e é informado para cada experimento.

Economia de espaço refere-se ao número de nós da árvore de derivação XML, sendo considerado adequado o número mínimo de nós para a codificação da informação geográfica segundo o padrão do OGC para cada serviço.

Economia de processamento refere-se ao número de invocações para requisição, obtenção do estado da requisição e obtenção do resultado final. É considerado adequado uma única invocação de cada tipo e é considerado negativo o aumento do número de invocações para obtenção do estado (pronto ou não pronto) da requisição.

Facilidade de implementação refere-se à necessidade de introdução de código adicional para desempenhar alguma tarefa. É considerado código adicional a criação de classes que não pertencem ao domínio da aplicação e são necessárias para, por exemplo, manipular dados no formato da GML, implementar um servidor HTTP, etc.

Facilidade de operação refere-se à necessidade de interação com o usuário. É considerado adequado o número mínimo de interação para cada transparência e inadequado a introdução de qualquer interface adicional com o usuário.

Flexibilidade de endereçamento trata da obrigatoriedade ou não de um endereço IP válido ou de um nome de domínio válido. É considerado adequada a não obrigatorie-

dade de um endereço IP válido.

Flexibilidade de escolha de protocolo de comunicação entre o cliente e o serviço refere-se a independência entre o cliente, os protocolos de comunicação e o serviço. A medida deste RNF é o número de protocolos capazes de implementar a comunicação, sendo considerado adequado o número três, que corresponde ao número de protocolos usados na avaliação.

Flexibilidade de operação refere-se ao grau de personalização do software-cliente para o uso de uma determinada cadeia de serviços. A medida é formulada pelo número de serviços que podem ser introduzidos em tempo de execução na cadeia de serviços.

Independência de provedor refere-se a independência entre o cliente e os provedores de serviços compostos (não padronizados). É considerado adequado que um cliente use um segundo provedor sem mudanças em seu próprio código.

Independência de rede refere-se a independência entre o cliente, os protocolos de comunicação e a rede de comunicação através da qual o cliente comunica-se com os serviços. Está diretamente relacionado com o *RNF flexibilidade de endereçamento* e está em nível adequado quando o cliente pode adotar um protocolo diferente daquele usado pelo serviço.

Independência de tecnologia refere-se a independência entre a tecnologia usada para implementar o software cliente e a tecnologia usada para implementar o serviço geoespacial. É considerado adequado quando a tecnologia de implementação em ambos é diferente.

Responsabilidade sobre workflow refere-se ao grau de participação do cliente no processo de encadeamento de serviços Web geoespaciais, tolerância e recuperação a falhas e troca de informação entre serviços usados. É medido pelo número de fluxos de informação de sentido serviçoX, cliente, serviçoY.

Velocidade de acesso refere-se à janela de tempo necessário para que um cliente reconheça que o resultado de sua requisição está disponível no serviço, em uma chamada assíncrona. É medido pelo tamanho da janela e está diretamente relacionado ao *RNF economia de comunicação*. O nível adequado é uma janela de tamanho 0, quando o cliente é notificado sobre a prontidão do resultado assim que o mesmo torna-se disponível no serviço.

Velocidade de recuperação refere-se ao número de invocações necessárias para que um cliente reinicie (e processe até o ponto de falha) ou esteja preparado para continuar o processamento de uma cadeia de serviços geoespaciais caso um dos componentes falhe. É considerado adequado o número de invocações necessárias para encontrar um serviço substituto ou uma réplica.

3.4.2 Transparência de Acesso

A transparência de acesso refere-se a aspectos tais como protocolo de comunicação de dados e formato de representação de dados trocados entre diferentes componentes de software através de uma rede de computadores. Os seguintes protocolos podem ser usados para serviços Web geoespaciais: HTTP sem SOAP [Whi05], HTTP com SOAP [Son04], SMTP [CPD06], UDP [Per03]. Os formatos de representação de dados são baseados em XML, sendo que normalmente são usados documentos GML [Whi05].

Os requisitos não funcionais (RNF) considerados foram *economia de comunicação*, *economia de espaço*, *economia de processamento*, *facilidade de implementação*, *flexibilidade de endereçamento*, *flexibilidade de escolha de protocolo de comunicação entre o cliente e o serviço*, *independência de provedor*, *independência de rede* e *velocidade de acesso*, os quais são apresentados na tabela 3.1 juntamente com as técnicas de satisfação implementadas. Na tabela, os símbolos + e − indicam impactos positivos e negativos, respectivamente, das técnicas de satisfação (colunas) sobre os RNF (linhas).

Em I1(ver figura 3.3), o serviço que faz a intermediação entre serviços primitivos e o cliente foi chamado de serviço *composto*, construído por um provedor ou integrador de serviços, enquanto que cada serviço da segunda camada, do OGC, foi chamado serviço *primitivo*. Na I2 (ver figura 3.4), só há serviços *primitivos* em conformidade com o OGC.

HTTP com GET O processo de invocação através do método GET do protocolo HTTP (sem SOAP), foi implementado assim como especificado pelo OGC para todos os serviços geoespaciais avaliados, com resposta em XML e GML [Whi05]. Ao *RNF economia de comunicação*, o protocolo HTTP com método GET oferece comunicação orientada a conexão, sem suporte a assincronicidade. Desse modo, quando o tempo necessário para resposta dos serviços primitivos para o cliente (em I2) foi aumentado, foi implementado um comportamento assíncrono de *invocar, aguardar, perguntar por prontidão e obter resposta quando pronto*. No entanto, como a atividade de perguntar por prontidão inicia-se pelo cliente, se percebeu um número de invocações intermediárias maior que 1, o que é negativo. Em I1, esse mesmo comportamento foi implementado entre o cliente e o integrador (serviço composto), quando obteve-se o mesmo resultado. A comunicação entre o serviço composto e os serviços primitivos em I1 não foi avaliada por não afetar o cliente com relação a este RNF.

Quando confrontado com o *RNF facilidade de implementação*, o uso de HTTP com GET exigiu codificação adicional nos clientes tanto no modelo I1 quanto I2, uma vez que a codificação e as restrições de tipo dos parâmetros passados na URI do serviço devem ser convertidos e verificados pelo cliente, além de essas propriedades serem definidas pelo

RNF	HTTP com GET	HTTP com SOAP	SMTP	UDP	XML/ GML	Integrador	CAS	CAS Gateway	
Economia de comunicação	-	-	+	+			+	+	
Economia de espaço		-							
Economia de processamento			+	+			+	+	
Facilidade de implementação	-	+	-	+	+	+	-	+	
Flexibilidade de endereçamento	+	+	-	-			-	+	
Flexibilidade de escolha de protocolo de acesso						+	-	+	
Independência de provedor	+	+	+	+		-	+	+	
Independência de rede				-					
Velocidade de acesso	-	-	+	+		+	+	+	

Tabela 3.1: RNF da transparência de acesso e técnicas de satisfação

desenvolvedor do software, o que é negativo.

O *RNF flexibilidade de endereçamento* no cliente foi suportado pelo protocolo HTTP com GET quando o cliente possuía e quando não possuía um endereço IP válido, o que deve-se ao fato de o HTTP ser orientado a conexão e as respostas serem encaminhadas através desta conexão, sem o uso do IP para o roteamento.

Além disso, uma vez que o protocolo HTTP sem SOAP é adotado como padrão pelo OGC, o *RNF independência de provedor* foi suportado, ao não impor qualquer necessidade de mudanças de código no software-cliente.

O *RNF velocidade de acesso* foi medido apenas em transações longas de serviços geoespaciais, quando foi emulada assincronicidade. Para isso, foi considerado adequado quando o cliente conhecia sobre a prontidão do resultado tão longo este tornava-se disponível no serviço e conseqüentemente possível de ser obtido pelo cliente. Com a implementação

de HTTP o cliente devia invocar o serviço para perguntar por prontidão, o que afeta os *RNF economia de comunicação* e *velocidade de acesso*, de modo que quando foi reduzido o número de invocações o *RNF velocidade de acesso* foi afetado negativamente; e mantê-lo no nível adequado apenas foi possível pelo aumento do número de invocações intermediárias, com os efeitos negativos já conhecidos para o *RNF economia de comunicação*

HTTP com SOAP Pela implementação do protocolo SOAP sobre HTTP [Son04], o *RNF economia de espaço*, que não havia sofrido nenhum impacto sem o protocolo SOAP, foi afetado negativamente pelo fato de este protocolo ter acrescentado pelo menos um nodo (do envelope SOAP) não associado com o esquema OGC na árvore de derivação em XML.

No entanto, o *RNF facilidade de implementação* foi afetado positivamente por nenhum código adicional ter sido introduzido no cliente uma vez que o *middleware* Apache Axis e a biblioteca padrão da linguagem de programação adotada foram suficientes. Por outro lado, a simples adoção de SOAP sobre HTTP não afetou outros requisitos importantes, como *velocidade de acesso* e *economia de comunicação*, tornando a maior facilidade de implementação a única diferença percebida.

SMTP A substituição do protocolo HTTP (síncrono) pelo protocolo SMTP (assíncrono), usado para correio eletrônico, foi experimentada sobre a proposta de Chung et al [CPD06] e em seguida pela construção de um servidor SMTP no cliente, o que favoreceu o *RNF economia de comunicação* ao reduzir o número de invocações intermediárias para 0, apesar de exigir ao cliente manter-se *online*.

O *RNF economia de processamento* foi então afetado positivamente uma vez que nenhum processamento intermediário de verificação de prontidão ocorreu.

No entanto, o *RNF facilidade de implementação* foi afetado negativamente pela necessidade de transformar o cliente em servidor SMTP, o que exigiu a implementação de código adicional não relacionado ao domínio da aplicação.

O *RNF flexibilidade de endereçamento* também foi afetado negativamente, pois o cliente como servidor SMTP exige o uso de um endereço IP válido, o que tornou-o disponível para o serviço remoto.

Finalmente, o *RNF velocidade de acesso* foi afetado positivamente, pois uma notificação de prontidão do serviço para o cliente foi encaminhada imediatamente após a disponibilidade da resposta, com vantagem sobre as técnicas de satisfação anteriores ao não afetar negativamente o *RNF economia de comunicação*.

UDP O segundo protocolo assíncrono implementado foi o UDP, o qual apresentou as mesmas propriedades do protocolo SMTP, explicado anteriormente. A única diferença foi ter afetado negativamente o *RNF independência de rede*, pelo fato de mensagens UDP exigirem um endereço IP válido para o roteamento, além de normalmente serem descartados por servidores do tipo *firewall*, por questões de segurança, no trânsito de mensagens da *extranet* para a rede local.

XML/GML A adoção de XML e sua linguagem derivada GML para codificação afetou positivamente o *RNF facilidade de implementação* ao não introduzir código adicional no cliente para troca de informação geográfica entre serviços e o cliente. Nos clientes, implementados nas linguagens Java [HC01a, HC01b] e PHP [AB03], foi possível usar apenas métodos de manipulação de XML disponíveis na biblioteca padrão das respectivas linguagens.

Integrador Especificamente na implementação 1 (I1), o papel de integrador foi avaliado e afetou positivamente o *RNF facilidade de implementação* ao não introduzir código adicional no cliente para comunicação entre os serviços primitivos e o cliente.

Adicionalmente, o *RNF flexibilidade de escolha de protocolo de acesso* para o cliente também foi impactado positivamente por permitir ao serviço composto responder ao cliente através de todos os protocolos avaliados (HTTP, SMTP e UDP), simultaneamente, e ao cliente receber resposta em qualquer destes protocolos, porém com prejuízo da universalidade do software-cliente, pois o *RNF independência de provedor* foi afetado negativamente devido a interface do serviço composto não ser padronizada.

O *RNF velocidade de acesso* pôde ser suportado em I1 através do uso dos protocolos SMTP e UDP, apenas, sem afetar o *RNF economia de comunicação* portanto.

CAS e CAS gateway A implementação e atendimento dos RNF em equilíbrio no cliente dá-se pelo grupo de funcionalidades denominado CAS, o qual também pode ser implementado em um proxy neutro denominado CAS *gateway*. O *RNF economia de comunicação* foi então afetado positivamente, pois o CAS emulou assincronicidade ao introduzir o comportamento de servidor HTTP no cliente (semelhante ao comportamento de servidor SMTP). Dessa forma, não houve invocação intermediária entre as invocações de pedido e obtenção de resposta. O CAS gateway, por sua vez, possibilitou implementar o envio de notificações por meio de um protocolo mais simples (UDP), transferindo para um proxy neutro a responsabilidade de receber notificações do serviço geoespacial, o que é viável em I1 e I2.

Conseqüentemente, os *RNF economia de processamento* (número de invocações intermediárias igual a 0), *facilidade de implementação* (permitiu usar UDP entre cliente e CAS gateway sem introdução de código adicional para implementar um servidor HTTP ou SMTP), *flexibilidade de endereçamento* (cliente pôde receber resposta assíncrona sem endereço IP válido) foram afetados positivamente.

O *RNF flexibilidade de escolha de protocolo de acesso* foi suportado de forma similar ao suporte dado pelo integrador, porém sem prejuízo para o *RNF independência de provedor*, uma vez que as funcionalidades são implementadas no cliente ou no gateway adotado pelo cliente, sendo esse gateway também independente dos provedores de serviços primitivos.

Finalmente, por emular assincronicidade e o cliente ser notificado sobre a prontidão da resposta imediatamente após sua disponibilidade no serviço, o *RNF velocidade de acesso* foi afetado positivamente.

3.4.3 Transparência de Localização

A transparência de localização refere-se a localização física dos diversos componentes de software que constituem o sistema de informação geográfico, o que é formado por clientes, serviços Web genéricos (WS), serviços Web geoespaciais (GWS), servidores de bancos de dados (BD), e outros.

Os requisitos não funcionais (RNF) considerados foram *economia de comunicação*, *economia de espaço*, *facilidade de implementação*, *facilidade de operação*, *flexibilidade de endereçamento* [cliente, serviço, BD], *flexibilidade de escolha de protocolo de comunicação entre o cliente e o serviço* e *independência de provedor*, os quais são apresentados na tabela 3.2 juntamente com as técnicas de satisfação implementadas.

Um componente importante para qualquer sistema distribuído é o serviço de rede *domain name system* (DNS), responsável pela tradução de nomes de domínio para endereços de Internet, ou endereços IP (Internet Protocol). Através do DNS, endereços IP podem ser alterados, o que é atualizado em tabelas DNS distribuídas. Este componente é essencial e sua indisponibilidade exige o uso do endereço IP do servidor em formato numérico. Entretanto, o DNS não participou de nossos experimentos, apesar de suportar os RNF flexibilidade de endereçamento [serviço], flexibilidade de endereçamento [BD], facilidade de implementação [cliente] e facilidade de operação [cliente], devido a possibilidade de se usar endereço IP dinâmico com um nome de domínio fixo, além da facilidade de configuração para o usuário.

IP A localização do cliente é importante no contexto dos SIG, especialmente quando este cliente solicita uma informação por meio de um protocolo não orientado a conexão

(como o UDP), e o servidor deve encaminhar a resposta para seu endereço IP. Com relação ao endereçamento IP, há duas possibilidades para o cliente: o cliente tem um IP válido, único e acessível na Internet; ou o cliente não tem um IP válido, isto é, o cliente pertence a uma rede privada (*intranet*) e acessa a rede pública através de um IP compartilhado por todos os clientes da rede privada. No primeiro caso, o cliente é acessível e tem as mesmas propriedades de servidores de rede. Enquanto segundo caso, o cliente é inacessível diretamente a partir da rede pública, e qualquer requisição realizada por aquele cliente tem a resposta interceptada por um componente da rede privada que faz o encaminhamento conforme políticas dessa rede, as quais podem aceitar ou rejeitar determinados serviços, protocolos e dados.

O segundo caso é o mais comum, o que afeta diretamente os RNF flexibilidade de endereçamento [cliente] e flexibilidade de escolha de protocolo de comunicação entre o cliente e o serviço [cliente]. No entanto, a primeira técnica de satisfação implementada foi a adoção de um IP válido para o cliente, através do qual o cliente pode receber notificações a partir de qualquer qualquer localização da Internet.

Primeiramente, foi observado que o *RNF economia de comunicação* foi afetado positivamente ao permitir que clientes e serviços fossem conhecidos pelo mesmo IP válido e estático sem que buscas em um catálogo fossem realizadas a cada invocação.

O *RNF economia de espaço*, por sua vez, foi afetado negativamente pela imposição de conhecer e armazenar réplicas e serviços substitutos estaticamente no código do software-cliente, embora a possibilidade de fazê-lo fosse desejável (identificadores estáticos de serviços).

Por outro lado, o *RNF facilidade de implementação* do cliente foi afetado de forma positiva ao favorecer o acesso ao cliente, por outros clientes ou pelos serviços, fazendo com que o cliente se tornasse provedor de parâmetros e *feedback*. Quando as respostas a requisições eram longas, foi implementado um padrão assíncrono de respostas o que foi favorecido pela existência de um endereço IP válido para o cliente. Isto também impactou positivamente o *RNF facilidade de implementação* do serviço, pois nenhum mecanismo de atualização de IP para as respostas assíncronas foi implementado, o que seria necessário quando o cliente entrava em modo de *espera* e quando retornava ao modo *online* receberia um endereço IP diferente do primeiro.

Porém, a implementação desta técnica de satisfação afetou negativamente o *RNF flexibilidade de endereçamento*, pois o funcionamento do cliente para obtenção de respostas assíncronas só foi possível com o uso de endereços IP válidos, e foi mais fácil pelo uso de endereços IP estáticos.

Com a atribuição de um endereço IP válido, estático ou dinâmico, todos os protocolos

analisados puderam ser implementados (HTTP, SMTP e UDP), o que afetou positivamente o *RNF flexibilidade de escolha de protocolo de acesso* no cliente e no serviço.

RNF	**IP**	**Bookmark**	**Catálogo**	**Integrador**	**CAS**	**CAS Gateway**	
Economia de comunicação	+	+	-	+			
Economia de espaço	-	-	+	+			
Facilidade de implementação	+	-	+	+	-	+	
Facilidade de implementação [serviço]	+				+	+	
Facilidade de operação		-	+	+			
Flexibilidade de endereçamento	-			+			
Flexibilidade de escolha de protocolo de acesso	+			+	-	+	
Flexibilidade de escolha de protocolo de acesso [serviço]	+			+	+	+	
Independência de provedor	+	+	+	-	+	+	

Tabela 3.2: RNF da transparência de localização e técnicas de satisfação

Finalmente, o *RNF independência de provedor* foi suportado pela implementação transparente de troca de mensagens através dos protocolos analisados, sem mudanças no software-cliente, especificamente na implementação 2.

Bookmark Com relação aos serviços Web e a complexidade de localização destes pelo cliente, foi usado o endereço estático (URI) de serviços diretamente no código-fonte do cliente, o que é denominado aqui como *bookmark*.

Os *RNF economia de comunicação*, *economia de espaço* e *independência de provedor* foram afetados de forma similar à produzida pela técnica de satisfação IP. No entanto, a implementação do carregamento do *bookmark* produziu efeito negativo no *RNF facilidade de implementação* do cliente, pela necessidade de implementar uma estrutura interna de dados para réplicas e implementar uma interface de atualização pelo usuário, o que também impactou negativamente o *RNF facilidade de operação*.

Catálogo O uso de catálogos de serviços foi outro mecanismo que auxiliou a localização de serviços, especialmente quando estes tinham nomes de domínio ou endereços IP dinâmicos, ou ainda quando o próprio cliente moveu-se de uma LSDI para outra e usou serviços de regiões diferentes. Neste último caso, o catálogo de serviços geográficos foi então usado para requisitar o IP de serviços de mesmo tipo, porém de LSDI diferentes, quando o cliente informava um retângulo envolvente diferente daquele no qual estava em requisições anteriores.

Isso permitiu a localização de serviços em tempo de execução, por meio de buscas no catálogo já padronizado, sem a participação de integradores e sem o armazenamento de endereços no interior do código do software-cliente (*RNF economia de espaço*, *facilidade de implementação*, *facilidade de operação* e *independência de provedor* foram afetados positivamente), porém a introdução de transações específicas para localização dos serviços de interesse afetou o *RNF economia de comunicação*.

Integrador Ao avaliar esta transparência na I1, o integrador de serviços se mostrou útil ao esconder os serviços primitivos do cliente e transferir para si a responsabilidade de localização destes serviços, ao contrário da técnica de catálogo. Isso afetou positivamente o *RNF economia de comunicação* e todos os outros anteriormente citados para a técnica de catálogo, com exceção do *RNF independência de provedor*, uma vez que o cliente não teve controle sobre os serviços primitivos usados pelo serviço composto. Adicionalmente, quando o servidor que possui o papel de integrador reside na mesma rede do cliente, não há problema de o cliente não possuir IP válido na Internet, o que também afetou positivamente o *RNF flexibilidade de endereçamento* [cliente].

CAS e CAS gateway Mesmo quando o integrador de serviços não participou da implementação, como no caso da I2, o serviço de catálogo de serviços foi adequado para localização dos serviços primitivos, assumindo parte do papel do integrador da I1. No entanto, algumas facilidades são desejáveis para a localização de clientes, especialmente no contexto da I2, onde é maior a importância de um IP válido da Internet. Assim,

são esperadas melhorias nos *RNF facilidade de implementação* [cliente, serviço] para suporte de respostas assíncronas, *RNF flexibilidade de endereçamento* [cliente], *flexibilidade de escolha de protocolo de acesso* [cliente, serviço] e *independência de provedor*. Esses requisitos foram atendidos pelo grupo de funcionalidades do *Client Access Service*.

O *RNF facilidade de implementação* do cliente foi afetado positivamente por um *proxy* neutro com as funcionalidades do CAS, através do qual notificações sobre a prontidão do serviço são enviadas para o cliente, as quais foram encaminhadas pelo *proxy* ao cliente mesmo quando o cliente mudou de endereço IP local, uma vez que o *proxy* local tem acesso privilegiado a tabela de endereços IP. O mesmo RNF para o serviço também foi afetado positivamente ao possibilitar respostas assíncronas e possibilitar reduzir as exigências de tempo para respostas a clientes com baixa capacidade de energia.

O *RNF flexibilidade de escolha de protocolo de acesso* para o cliente foi afetado positivamente pela atribuição da responsabilidade de intermediação ao *proxy*, enquanto a comunicação entre o *proxy* neutro e o cliente tornou-se mais flexível, sendo possível implementar protocolos mais simples como o UDP e até mesmo protocolos mais específicos como o SMS (*Short Messages System*), das redes de telefonia móvel celular.

Finalmente, o *RNF independência de provedor* foi afetado positivamente (ao contrário do produzido pela técnica de Integrador), por deixar as responsabilidades de intermediação do integrador em um proxy neutro possível de ser padronizado e fornecer serviço a múltiplos provedores de serviço Web geoespacial.

3.4.4 Transparência de Migração

Esta transparência refere-se a capacidade de relocação de componentes de software sem que complexidade seja adicionada nos próprios componentes.

Os requisitos não funcionais (RNF) considerados foram *economia de comunicação*, *facilidade de implementação* e *independência de provedor*, os quais são apresentados na tabela 3.3 juntamente com as técnicas de satisfação implementadas.

RNF	HTTP	UDP	Integrador	DXS	
Economia de comunicação	+	+	+	+	
Facilidade de implementação	-	-	-	+	
Independência de provedor	+	+	-	+	

Tabela 3.3: RNF da transparência de migração e técnicas de satisfação

HTTP A implementação do protocolo de comunicação HTTP com SOAP em processos longos exigiu invocações em duas chamadas, a primeira para requisição e a segunda para obtenção do resultado. No entanto, para reduzir o número de invocações com objetivo de conhecer o estado de prontidão do resultado, foi implementado o envio de notificação do serviço para o cliente.

Na I2, sem intermediação do integrador, o serviço informava o cliente através de transações HTTP dirigidas ao cliente (cliente implementado como servidor HTTP). Porém, com a motivação da alteração de endereço IP do cliente (cliente em movimento ou cliente em modo de *espera*), o cliente foi obrigado a informar o novo endereço IP ao serviço, com apenas uma invocação a cada troca de endereço, o que afetou positivamente o *RNF economia de comunicação*.

No entanto, o *RNF facilidade de implementação* foi afetado negativamente pela necessidade de controle pelo cliente sobre cada transação, com introdução de código adicional não relacionado ao domínio da aplicação.

O *RNF independência de provedor* foi afetado positivamente pela não necessidade de um controle centralizado de endereço IP do cliente para notificação aos serviços em uso (em processamento) pelo cliente.

UDP A substituição do protocolo HTTP pelo protocolo UDP não retirou do cliente a necessidade de informar ao serviço em processamento a mudança de endereço, uma vez que a notificação (agora em UDP) era redirecionada para o endereço IP que o cliente tinha no momento da primeira invocação. Conseqüentemente, nenhuma mudança nos RNF foi percebida.

Integrador Em nosso contexto, os componentes que migraram foram os clientes, o que muitas vezes já é tratado por protocolos específicos nas redes de GPRS (*General Packet Radio Services*) da telefonia móvel, com transferência de pacotes até o dispositivo móvel por meio de um controle centralizado na CCC (Central de Comutação e Controle). Esta tecnologia não foi avaliada, apesar do padrão de controle ter sido emulado na I1, através do Integrador.

O único resultado percebido, com relação às técnicas anteriores, foi um impacto negativo no *RNF independência de provedor*, devido a necessidade de introdução de código não padronizado nos componentes de software.

DXS A manutenção do *RNF economia de comunicação* e o suporte do *RNF independência de provedor* foi garantido por meio de um mecanismo neutro, o DXS, para o

qual respostas são encaminhadas a partir dos serviços, e de onde são obtidas as respostas pelo cliente.

O *RNF facilidade de implementação* do cliente também foi afetado positivamente ao não introduzir qualquer código no software-cliente, exigindo apenas invocações em duas chamadas, a primeira para requisição e a segunda para obtenção do resultado diretamente no DXS.

3.4.5 Transparência de Relocação

Esta transparência refere-se a relocação de interfaces de serviços usadas por outros serviços ou por clientes finais, isto é, refere-se a invocação de serviços que migraram de um local para outro sem mudanças substanciais nos clientes destes serviços.

Os requisitos não funcionais (RNF) considerados foram *economia de comunicação*, *economia de processamento*, *facilidade de implementação*, *independência de provedor* e *independência de tecnologia* [cliente, serviço], os quais são apresentados na tabela 3.4 juntamente com as técnicas de satisfação implementadas.

A migração de serviços não é prevista pelo OGC, embora seja prevista neste trabalho por meio de aplicações para os serviços *Data Exchange Service* e *Client Access Service* , através dos quais clientes podem ser usados como provedores de informação geográfica para outros clientes e mesmo para serviços que coletam informação em tempo real. Esta transparência muitas vezes depende da transparência de replicação, na próxima seção.

RNF	**Acesso Direto**	**Integrador**	**CAS Gateway**	
Economia de comunicação	-	+	-	
Economia de processamento	-	+	-	
Facilidade de implementação	-	+	-	
Independência de provedor	+	+	+	
Independência de tecnologia	-	+	+	

Tabela 3.4: RNF da transparência de relocação e técnicas de satisfação

Duas possibilidades de relocação, neste sentido, se relacionam com os dados (relocação da fonte de dados) e com funções (relocação de serviços). Em nosso cenário, a relocação é

materializada principalmente em balanceamento de carga, quando o cliente é *redirecionado* transparentemente para outro servidor de dados ou funções, seja este servidor uma réplica ou substituto do serviço requisitado. Uma terceira possibilidade de relocação se relaciona com o movimento da fonte de dados, no caso de provedores de dados móveis tais como redes de geosensores sem fio e clientes móveis usados como provedores de informação em tempo real.

Acesso Direto O terceiro caso, relacionado ao cliente móvel como *serviço* de informação geográfica, exige que o novo endereço IP seja informado para outros clientes e serviços através do envio de mensagens, sendo que, dispensado o integrador na I2, uma mensagem é enviada para cada mudança de endereço IP, o que afetou o *RNF economia de comunicação* para o cliente real.

O *RNF economia de processamento* também foi afetado negativamente devido a necessidade de alterar o endereço físico do serviço no cliente para as próximas invocações. Adicionalmente, isso produziu efeito negativo nos *RNF facilidade de implementação*, em função da introdução de código específico para atualização de IP, e *independência de tecnologia*, devido a necessidade de comunicação assíncrona para notificação aos clientes e a conseqüente imposição de alguns protocolos ou funcionalidades ao cliente.

No entanto, o *RNF independência de provedor* foi afetado positivamente na I2, pela não participação do integrador da I1, apesar de não haver interface padronizada de notificação ao cliente sobre mudanças de endereço do serviço.

Integrador Na I2, atividades de mudança de endereço dos serviços primitivos foram atribuídas integralmente ao integrador, as quais tornaram-se transparentes para o cliente. Neste contexto, o serviço continua a acessar o serviço primitivo através do serviço composto, ou ainda através de um "apelido" dado àquele serviço pelo integrador. Toda alteração de endereço é informada ao integrador, sem trocas de mensagens entre o serviço primitivo e o cliente, o que afetou positivamente os *RNF economia de comunicação* e *economia de processamento*.

Como nenhuma informação é dada ao cliente e nenhuma mudança de comportamento é exigida do cliente, os *RNF facilidade de implementação* e *independência de tecnologia* são igualmente afetados de forma positiva.

Finalmente, ao implementar o integrador como um proxy neutro, através do qual os serviços primitivos são acessados indiretamente por meio de um "apelido" fixo, comportamento este que pode ser implementado no integrador ou até mesmo no catálogo de serviços, o *RNF independência de provedor* também foi afetado positivamente, pois a

presença do integrador também se tornou transparente ao cliente, sem afetar os demais RNF.

De volta à I2, foi implementado um *proxy* com as funcionalidades do *Client Access Service*, sendo que as responsabilidades de relocação retornaram para o cliente. Isso afetou os *RNF economia de comunicação*, *economia de processamento* e *facilidade de implementação* pelas mesmas razões impostas pela técnica de Acesso Direto.

A única vantagem desta técnica sobre a de Acesso Direto foi o efeito positivo sobre o *RNF independência de tecnologia*, uma vez que o CAS retirou as restrições de protocolo para o recebimento de mensagens assíncronas do serviço para o *proxy*, e deste para o cliente.

No entanto, é possível concluir que o papel de integrador em I2 (como um *proxy* neutro) é superior as outras técnicas e apenas não se aplica em certos contextos, tais como aqueles nos quais são usados apenas clientes gordos.

3.4.6 Transparência de Replicação

A transparência de replicação refere-se a criação de réplicas consistentes para aumentar o desempenho e a disponibilidade do sistema. Podem ser replicados os dados ou os bancos de dados geográficos (BD), os servidores físicos de um mesmo provedor, ou serviços geográficos em provedores diferentes, sendo que estes últimos não devem ser necessariamente consistentes mas apenas equivalentes. Podem ainda ser replicados serviços de infra-estrutura tais como catálogos de serviços e o serviço *Domain Name System* (DNS) para atribuição de endereços de Internet a domínios.

Os requisitos não funcionais (RNF) considerados foram *economia de comunicação*, *economia de processamento*, *facilidade de implementação*, *facilidade de operação* e *independência de provedor*, os quais são apresentados na tabela 3.5 juntamente com as técnicas de satisfação implementadas.

Integrador A transferência de responsabilidade sobre réplicas para o provedor de serviços compostos (caso da I1) foi suficiente para assegurar o suporte aos *RNF economia de comunicação*, *economia de processamento*, *facilidade de implementação* e *facilidade de operação*, pois o controle sobre falhas e adoção de réplicas foi atribuído integralmente ao integrador e tornou-se transparente para o cliente, apesar de prejudicar o *RNF independência de provedor*, uma vez que não havia mecanismos de verificação sobre quais serviços foram usados pelo integrador e a quais fornecedores tais serviços pertencem. Isto é particularmente importante para definir qualidade e consistência das réplicas, o que torna-se possível pelo acesso a informações do catálogo ou pelo acesso direto aos serviços.

RNF	Integrador	Bookmark	Catálogo	TCS	
Economia de comunicação	+	+	-	-	
Economia de processamento	+	-	+	-	
Facilidade de implementação	+	-	+	+	
Facilidade de operação	+	-	+	+	
Independência de provedor	-	+	+	+	

Tabela 3.5: RNF da transparência de replicação e técnicas de satisfação

Bookmark Ao retirar o papel de integrador, foi atribuído ao cliente o papel de invocar réplicas em caso de indisponibilidade. No entanto, o endereço de serviços não indica réplicas uniformemente e o conhecimento sobre sua existência e as URI correspondentes deviam ser conhecidas antecipadamente.

Um modelo de *bookmark* interno ao software-cliente foi implementado e afetou negativamente os *RNF economia de processamento*, devido a necessidade de controle de falhas e carregamento de interfaces de réplicas, *facilidade de implementação*, devido a introdução de URI de réplicas no código do software-cliente e necessidade de criação de interface de atualização com o usuário, e também o *RNF facilidade de operação*, nos casos em que URI mudavam e o usuário devia fazer atualizações.

O *RNF independência de provedor*, por sua vez, foi afetado positivamente devido ao acesso direto ao catálogo, aos serviços e seus respectivos serviços substitutos e réplicas.

Catálogo Ao contrário do *bookmark*, o catálogo também foi usado como fonte de informação sobre réplicas, o que mostrou-se negativo para o *RNF economia de comunicação*. Porém, todos os outros RNF foram afetados positivamente através da busca por réplicas no catálogo, remotamente e da não necessidade de introduzir réplicas no código do software-cliente nem de criar interface de atualização para o usuário.

TCS Implementar mecanismos mais robustos de uso de réplicas sem prejudicar requisitos tais como economia de comunicação, economia de processamento e independência de provedor é desejável. Além disso tais requisitos impactam diferentemente em diferentes contextos e dispositivos, o que impede que uma configuração única seja a ideal sempre.

Isto pode ser atendido automaticamente pelo *Transaction Control Service* o qual permite configurar tais requisitos para diferentes contextos e dispositivos.

O cliente recebeu quatro códigos compatíveis com as técnicas Integrador, *Bookmark* e Catálogo, e o quarto código foi responsável por combinar os RNF para favorecê-los em uma distribuição diferente daquelas criadas pelas técnicas de satisfação anteriores. Pelo código construído, a busca em um catálogo foi realizada em caso de falhas, o que impactou negativamente o *RNF economia de comunicação*, deixou o tratamento de falhas a cargo do cliente, o que afetou negativamente o *RNF economia de processamento*, permitiu atualização da lista de serviços substitutos no cliente através do catálogo e introduziu o quarto código automaticamente no software-cliente, o que favoreceu os *RNF facilidade de implementação* e *facilidade de operação*, além de ter mantido o acesso direto aos serviços (sem integrador) para o cliente, o que afetou de forma positiva o *RNF independência de provedor*.

3.4.7 Transparência de Persistência

Esta transparência refere-se a criação e uso de objetos no lado do servidor, o que principalmente melhora o desempenho de espaço e processamento em componentes distribuídos de software.

Os requisitos não funcionais (RNF) considerados foram *economia de comunicação* [cliente, serviço], *economia de espaço* [cliente, serviço], *economia de processamento*, *facilidade de implementação* [cliente, serviço], *independência de provedor* e *independência de tecnologia* [cliente, serviço], os quais são apresentados na tabela 3.6 juntamente com as técnicas de satisfação implementadas.

As transparências e suas técnicas de satisfação adotadas muitas vezes interferem umas nas outras. Este é o caso do suporte a persistência no lado do servidor para uma maior tolerância a falhas. Para isto, é essencial que o estado das transações seja armazenado em diferentes componentes distribuídos pela rede, especialmente na presença de meios de comunicação instáveis ou caros.

Como ilustração, se uma operação qualquer falha após uma seqüência longa de operações no interior de uma cadeia de serviços, o cliente deve rastrear o serviço que falhou, selecionar uma réplica e invocá-la dinamicamente sem o reenvio de dados, que já estão em algum meio remoto, ou a inicialização de toda a cadeia de serviços.

Framework Mas isto não é trivial de se implementar, uma vez que as incompatibilidades entre serviços OGC e serviços W3C dificultam o uso de mecanismos prontos de persistência oferecidos por servidores de aplicação (.NET ou J2EE, por exemplo). Porém,

como tais mecanismos são projetados para servidores de aplicação específicos, é uma hipótese válida que alguns deles possam prejudicar o *RNF independência de tecnologia* [cliente, serviço].

Com isso, o *RNF economia de comunicação* é afetado negativamente, pois serviços não podem acessar outros serviços incompatíveis para recuperar dados persistentes, e na I2, com acesso direto, o cliente é obrigado a armazenar uma identificação para recuperação do dado persistente, em tipo diferente quando comparadas diferentes tecnologias, o que também afeta negativamente o *RNF economia de espaço*.

Como uma parcela de tarefas é transferida para o lado do servidor, a adoção de um *framework* favorece os *RNF economia de processamento* e *facilidade de implementação* do cliente e do serviço, apesar de afetar negativamente o *RNF independência de provedor*, uma vez que cada provedor pode adotar uma tecnologia diferente.

RNF	**Framework**	**Integrador**	**Serviço OGC**	**DXS**	
Economia de comunicação [cliente, serviço]	-	+	-	+	
Economia de espaço [cliente, serviço]	-	+	+	+	
Economia de processamento	+	+	+	+	
Facilidade de implementação [cliente]	+	+	+	+	
Facilidade de implementação [serviço]	+	+	+	+	
Independência de provedor	-	-	+	+	
Independência de tecnologia [cliente]	-	+	+	+	
Independência de tecnologia [serviço]	-	+	+	+	

Tabela 3.6: RNF da transparência de persistência e técnicas de satisfação

Integrador A I1 concentrou responsabilidades por persistência no integrador, afetando de forma positiva o *RNF independência de tecnologia*, mas com conseqüente impacto ne-

gativo no *RNF independência de provedor*, uma vez que o OGC não definiu interfaces para objetos remotos. Foram então atribuídos identificadores para transações, possibilitando solicitar o reenvio de uma resposta já recebida e solicitar uma resposta assíncrona através de uma segunda chamada (transações longas), através do fornecimento do identificador da primeira invocação.

Esse procedimento afetou positivamente os *RNF economia de comunicação*, *economia de espaço*, apesar da responsabilidade de armazenar os identificadores, *economia de processamento*, possível pela capacidade de reuso de dados previamente processados e não concluídos devido a ocorrência de falhas, *facilidade de implementação*, por implementar persistência totalmente do lado do integrador, e finalmente *independência de tecnologia*, ao deixar apenas a responsabilidade de armazenamento de identificadores de transação para o cliente.

Serviço OGC Na I2 foi também experimentado deixar a responsabilidade de persistência para o serviço, o que já existe para o serviço de roteamento, através do qual é possível recuperar trechos da rota após se ter percorrido alguma parte do percurso.

Todo esforço na I2 fez com que a responsabilidade do *workflow* fosse atribuída ao cliente, equiparando esta técnica à técnica *Framework*, exceto pelo *RNF independência de tecnologia* no cliente, o qual foi afetado positivamente por criar a possibilidade do controle de persistência de dados através de simples identificadores de transação.

DXS Em seguida, tal recurso foi movido dos serviços primitivos para um serviço genérico, responsável por receber respostas de serviços primitivos, armazená-las e encaminhá-las ao cliente sempre que solicitado, o que é suportado pelo *Data Exchange Service*. Isto se mostrou adequado para todos os RNF, inclusive para o *RNF economia de comunicação* [cliente] no contexto da I2, o qual ainda não havia sido afetado positivamente naquele modelo de computação, ao permitir o encadeamento de serviços em um *workflow* através do qual a resposta de um serviço serve como parâmetro para o próximo da cadeia de serviços.

3.4.8 Transparência de Falha

Esta transparência refere-se a recuperação do sistema e suporte a disponibilidade caso falhas ocorram nos componentes que atendem o sistema, tais como falhas nas redes de computadores, paradas de software, negação de acesso a serviço, e outros.

Os requisitos não funcionais (RNF) considerados foram *auditabilidade de componentes*, *economia de comunicação*, *economia de espaço*, *economia de processamento*, *facilidade*

de implementação, *independência de provedor* e *velocidade de recuperação*, os quais são apresentados na tabela 3.7 juntamente com as técnicas de satisfação implementadas.

RNF	Integrador	Bookmark	Catálogo	TCS	DXS	
Auditabilidade de componentes	-	+	+			
Economia de comunicação	+	-	-		+	
Economia de espaço	+	-	-		+	
Economia de processamento		-	+	+	+	
Facilidade de implementação	+	-	+	+	+	
Independência de provedor	-	+	+	+	+	
Velocidade de recuperação		-	-		+	

Tabela 3.7: RNF da transparência de falha e técnicas de satisfação

A implementação de tolerância a falhas, fator importante para viabilizar o uso de sistemas de informação geográficos no suporte a aplicações de missão crítica, é garantida por meio de serviços OGC tanto em I1 quanto em I2.

Integrador No entanto, na I1 o integrador é responsável pela implementação de tolerância a falhas, sem qualquer mecanismo de auditoria (o que é obrigatório: *RNF auditabilidade de componentes*) para o cliente. Assim, o cliente não tem como validar, testar e comparar dois provedores no aspecto de tolerância a falhas. Apesar disso, os *RNF economia de comunicação*, *economia de espaço* e *facilidade de implementação* foram beneficiados pela transferência de responsabilidade do cliente para o Integrador. Finalmente, o principal RNF prejudicado foi a *independência de provedor*, uma vez que qualquer discordância com as políticas de tolerância a falhas só pode ser resolvida pela adoção de outro provedor, o que se mostra difícil por ausência de padronização das interfaces de serviços compostos.

Bookmark Ao transferir a responsabilidade sobre a tolerância a falhas para o cliente, o controle de réplicas e substitutos foram primeiramente armazenados no cliente estaticamente, através da técnica *bookmark*. A simples transferência favoreceu os *RNF auditabilidade de componentes*, ao permitir acesso aos metadados dos serviços, e *independência*

de provedor, pois foram usadas apenas interfaces padronizadas dos serviços primitivos.

No entanto, quando serviços foram induzidos a falhar, os *RNF economia de comunicação*, *economia de espaço*, *economia de processamento* e *velocidade de recuperação* foram impactados negativamente, pois o cliente ocupou-se do reenvio dos parâmetros originais ao reiniciar o *workflow*, do armazenamento dos parâmetros e resultados intermediários já processados para controle do *workflow* e do processamento e transmissão das informações intermediárias para evitar a reinicialização do *workflow*. Adicionalmente, o *RNF facilidade de implementação* também foi afetado negativamente, pela necessidade de definir estaticamente os serviços substitutos e réplicas.

Catálogo A introdução do catálogo de serviços possibilitou encontrar serviços substitutos ou réplicas dinamicamente, o que favoreceu o *RNF facilidade de implementação*, o que permitiu selecionar novos serviços a partir da região geoespacial à qual estão associados e das funcionalidades oferecidas pelos serviços primitivos, sem contudo eliminar a necessidade de reenvio de informação e controle do *workflow*.

TCS e DXS Assim, um serviço baseado no *Data Exchange Service* foi usado para eliminar a necessidade de reenvio de informação em caso de falhas, quando parâmetros e resultados ficaram armazenados no servidor DXS. Isso afetou positivamente os *RNF economia de comunicação*, *economia de espaço* e *economia de processamento*, além do *RNF velocidade de recuperação* pela não necessidade de reenvio dos parâmetros originais e da possibilidade de recuperar-se de falhas sem a reinicialização de todo o *workflow*. Toda a complexidade associada ao *workflow* e o DXS foi gerada automaticamente como código executável pelo cliente através do serviço TCS, o qual finalmente favoreceu os *RNF economia de processamento* e principalmente *facilidade de implementação*.

3.4.9 Transparência de Transação

Esta transparência refere-se a coordenação de tarefas entre grupos de componentes de software, o que atribui uma semântica de processo (*workflow*) baseada na ordem das invocações, mesmo no caso de falhas ou na ocorrência de decisões do usuário via interações com o sistema.

Os requisitos não funcionais (RNF) considerados foram *controle sobre componentes*, *economia de comunicação*, *economia de processamento*, *facilidade de implementação*, *flexibilidade de operação*, *independência de provedor* e *responsabilidade sobre workflow*, os quais são apresentados na tabela 3.8 juntamente com as técnicas de satisfação implementadas.

Integrador Na I1, toda a coordenação do *workflow* de um serviço geográfico ficou sob a responsabilidade do integrador de serviços, apesar de ainda ser permitido ao software-cliente reunir cadeias de serviço (de um ou mais integradores) em um *workflow* do cliente.

Essa retirada de responsabilidade do cliente para o integrador (em I1) afetou negativamente o *RNF controle sobre componentes*, uma vez que não há uma interface padronizada para acesso indireto aos metadados dos serviços primitivos. No entanto, essa situação beneficiou o *RNF economia de comunicação*, ao reduzir para 0 o número de invocações do tráfego intermediário entre a primeira invocação e a obtenção do resultado final, o *RNF economia de processamento*, ao deixar para o integrador toda a responsabilidade sobre o processamento intermediário, e beneficiou o *RNF facilidade de implementação*, ao retirar código do software-cliente e equipará-lo a categoria de "cliente magro".

Por outro lado, o *RNF flexibilidade de operação* foi afetado negativamente devido ao fato de o cliente não ser capaz de redefinir o *workflow* segundo suas necessidades. O *RNF independência de provedor* foi afetado pela mesma propriedade, pois quaisquer mudanças de *workflow*, qualidade de serviço ou outra característica exigiu a mudança de serviço composto, de outro integrador. Finalmente, o *RNF responsabilidade sobre workflow* foi afetada negativamente pelo fato de o cliente não foi capaz de testar diferentes *workflows* e assegurar atributos importantes como privacidade e tolerância a falhas.

RNF	Integrador	TCS	DXS	
Controle sobre componentes	-	+		
Economia de comunicação	+	-	+	
Economia de processamento	+	-	+	
Facilidade de implementação	+	+	+	
Flexibilidade de operação	-	+		
Independência de provedor	-	+		
Responsabilidade sobre workflow	-	+		

Tabela 3.8: RNF da transparência de transação e técnicas de satisfação

TCS e DXS Ao atribuir as responsabilidades citadas ao cliente, o impacto sobre os RNF é exatamente o oposto, o que é negativo para os *RNF economia de comunicação*

e *economia de processamento*, requisitos essenciais para os clientes magros e ricos, e é também negativo para o *RNF facilidade de implementação*. O equilíbrio de responsabilidades entre os lados de cliente e de servidor só é possível com a adoção conjunta dos serviços *Transaction Control Service* e *Data Exchange Service*, sendo que a parcela de contribuição de cada um é mostrada na matriz de adjacência representada na tabela 3.8, onde o DXS melhorou os *RNF economia de comunicação*, *economia de processamento* e *facilidade de implementação*, ao manter os resultados preliminares ou intermediários no lado do servidor, em um repositório temporário, ao evitar que manipulação de fluxos de dados extensos acontecessem no cliente e finalmente ao facilitar o encaminhamento de respostas de um serviço para outro serviço, o qual os recebe como parâmetros. O TCS, por sua vez, resolveu as limitações encontradas anteriormente nos *RNF flexibilidade de operação*, *independência de provedor* e *responsabilidade sobre workflow*, ao facilitar o desenvolvimento de clientes no contexto de I2 e introduzir código do DXS automaticamente.

3.4.10 Avaliação: Transparências no Desenvolvimento de Clientes

A implementação 1 (I1), descrita na seção 3.2.2, impõe o uso de um provedor conhecido que intermedeia a comunicação entre o software-cliente e os vários serviços e servidores que compõem a LSDI. Por outro lado, a implementação 2 (I2), descrita na seção 3.2.3, não usa esse provedor conhecido e obriga o cliente a acessar diretamente os vários serviços e servidores, bem como a processar toda a informação geográfica obtida durante as requisições.

Com essas diferenças, o primeiro padrão tornou simples o desenvolvimento de clientes e servidores com requisitos de engenharia importantes. No entanto, o mesmo padrão introduziu dependência excessiva dos clientes em relação a provedores de serviços, especialmente em situações onde não há informação sobre os recursos (próprios e de terceiros) usados pelos provedores.

Por outro lado, o segundo padrão de distribuição tornou possível ao cliente o acesso aos metadados das fontes originais. No entanto, essa abordagem exigiu o abandono do papel de "integrador de serviços", do qual se ocupa o provedor no suporte aos RNF economia de comunicação e economia de processamento, principalmente. De uma maneira geral, o padrão causou, para o cliente, um aumento dos custos de tráfego e processamento de dados intermediários, uma vez que o cliente tornou-se responsável por requisições a vários serviços e, uma vez que ele recebe as respectivas respostas, algumas operações sobre estes dados intermediários são necessárias antes do encaminhamento destes como entrada para

os próximos serviços da cadeia de serviços. Mas este padrão também possibilitou monitorar e definir parâmetros e requisitos de engenharia no lado do cliente e, conseqüentemente, aumentou a independência e do cliente em relação aos provedores de serviço e permitiu a adequação do *workflow* a capacidade do dispositivo onde o cliente é executado.

Modularização

Funcionalidade e dados são encapsulados em serviços Web diretamente em seus provedores ou fornecedores originais. No entanto, percebeu-se que a I1 exigiu a criação de tantos serviços não-OGC quantas fossem as necessidades urbanas específicas apresentadas. Dessa maneira, novas especializações de serviço tornaram-se difíceis e fortemente dependentes de uma solução sem possibilidade de atendimento das especificações do OWS e OpenLS (Perspectiva *Tecnológica* do ORM). Isso é resultado da impossibilidade de se construir um cliente magro que invoque diretamente os serviços do OGC, por estes serviços imporem um tráfego intermediário maior do que estes clientes são capazes de manipular, como descrito na transparência de acesso, seção 3.4.2.

Porém, a construção de sistemas modulares no provedor (em I1) é garantida pela implementação e invocação de serviços assim como especificados pelo OGC, pois o integrador de serviços tem maior flexibilidade de implementação que um cliente magro em I2. Por outro lado, um cliente gordo em I2 pode manter a propriedade de modularidade em um nível adequado ao não possuir as limitações presentes no cliente magro, e neste caso tem capacidade comparável a de um provedor.

Personalização de Sistemas

Há dois usuários possíveis para um serviço Web geoespacial: desenvolvedores de novos serviços (enfatizado em I1) e desenvolvedores de softwares-cliente (enfatizado em I2), o que representou o principal foco desta pesquisa.

Desenvolvedores implementam softwares-cliente pela invocação de serviços Web suportados por provedores de serviço ou pela invocação de serviços primitivos do OGC diretamente, sem intermediários. No primeiro caso, percebeu-se a dificuldade de garantir a universalidade do software-cliente, uma vez que os serviços não são padronizados em seu nível conceitual (perspectiva Organizacional do ORM).

No segundo caso, é possível garantir a universalidade do software-cliente, mas com dificuldades no mapeamento da perspectiva de *Engenharia* para a perspectiva *Tecnológica*, isto é, na implementação de componentes de software adequados para a variedade de aplicações-cliente e dispositivos-cliente, especialmente no contexto urbano dos serviços de

localização.

Dessa forma, há dificuldades na personalização de sistemas concomitante à garantia de universalidade do cliente de acesso a informação geográfica e à transparência no desenvolvimento destes para a grande variedade de aplicações geográficas demandadas hoje.

Controle de Exceções

Serviços compostos ou clientes gordos tendem a usar somente serviços primitivos, o que facilita o controle de exceções e permite ao serviço ou cliente saber qual serviço falhou. Ao mesmo tempo, torna-se possível a seleção de réplicas para a substituição do serviço indisponível.

Entretanto, a rastreabilidade de falhas em tempo de execução pelo cliente é comprometida na I1, quando o controle de exceções é realizado integralmente pelo provedor de serviços invocado, e não pelo cliente. Isto tem impacto sobre a validação de indicadores de tolerância a falhas certificados pelos provedores, e impede o cliente de obter resultados parciais da operação, como visto na transparência de falha.

Adicionalmente, caso o cliente opte por usar uma réplica do provedor indisponível, o reenvio dos parâmetros originais é obrigatório, mas estes nem sempre estão disponíveis (por exemplo, no caso de uma rede de sensores geográficos sem fio) ou o reenvio pode ter custo de comunicação elevado (por exemplo, no caso de clientes móveis).

Encadeamento Dinâmico de Serviços

A invocação de réplicas, serviços concorrentes e serviços compatíveis de outra LSDI é possível para serviços compostos (na I1) e para clientes (na I2). Porém, na I1, clientes dependem de serviços compostos que definem a sua cadeia de serviços internamente, sem controle e mecanismos de monitoramento por parte do cliente. Enquanto que na I2, clientes magros são prejudicados pela necessidade da recepção de respostas dos serviços para o encaminhamento destes aos próximos serviços da sua cadeia. Em se tratando de serviços Web geoespaciais, isso significa a retransmissão de um volume significativo de dados.

Logo, a substituição do papel do serviço composto por um mecanismo de criação dinâmica de cadeias de serviço é útil por permitir aos clientes de diferentes tipos acessarem serviços geográficos de diferentes formas, aproveitando as vantagens dos provedores de serviços compostos e levando responsabilidade sobre um conjunto expressivo de aspectos para o cliente.

Suporte a Transações Longas

Outro aspecto tecnológico (perspectivas de *Engenharia* e *Tecnológica*) que limita o projeto de sistemas de informação reais é a falta de suporte a comunicação assíncrona, quando o protocolo HTTP é usado.

Isto é particularmente sério no contexto de clientes móveis, para os quais o custo de comunicação e energia é maior.

Clientes Universais para LSDI

Todos os aspectos citados anteriormente conduzem a um equilíbrio entre os benefícios e dificuldades decorrentes da adoção da I1 ou da I2, portanto tornando difícil uma escolha direta. Uma conseqüência disso no mundo real é a predominância de serviços específicos para dispositivos específicos, além da adoção de interfaces não padronizadas para adição de recursos necessários a aplicações geográficas comuns.

Predominam, no mercado atual, softwares geográficos proprietários (para diversos dispositivos e para um conjunto sempre limitado de regiões), e verifica-se a dificuldade de se implementar novos softwares, principalmente softwares-cliente, para toda e qualquer LSDI, em conformidade com os padrões do OGC [UCF+05].

Através dos serviços de infra-estrutura propostos na próxima seção, pretende-se implementar as alterações julgadas necessárias nos serviços primitivos OGC e nos serviços compostos não OGC, com o objetivo de dar suporte às propriedades e aos RNF citados anteriormente.

Capítulo 4

Serviços de Infra-estrutura Projetados

Dadas as necessidades de aplicações urbanas de informação geográfica, assim como antecipadas no capítulo anterior, e as principais limitações de serviços Web geoespaciais especificados pela OGC, foram desenvolvidos como parte deste trabalho alguns serviços de infra-estrutura que solucionam as limitações de engenharia encontradas.

Estes serviços foram construídos a partir de um processo de desenvolvimento em espiral [Boe98] da implementação 2 (Seção 3.2.3), de modo que funcionalidades de serviços em conformidade com a OGC eram implementadas na medida em que se mostravam necessárias no cenário de uso, juntamente com serviços complementares que representassem funções e modelos não suportados por serviços padronizados.

O primeiro serviço, denominado *Data Exchange Service*, é usado para suporte de persistência entre clientes e servidores. O segundo, denominado *Client Access Service*, é usado para troca de informação e controle de acesso direto a clientes por servidores e outros clientes. O terceiro, denominado *Transaction Control Service*, é usado por clientes e servidores para melhorar aspectos de engenharia, tais como recuperação em caso de falhas, criação dinâmica de cadeias de serviço, definição de *workflow* em alto-nível e outros. Esses serviços serão detalhados nas seções a seguir.

4.1 Data Exchange Service

O *Data Exchange Service* (DXS) suporta comunicação entre clientes e servidores, entre servidores (comunicação de servidor para servidor) e entre clientes (comunicação de cliente para cliente). O comportamento do DXS é análogo a uma memória temporária, onde dados podem ser armazenados e recuperados livremente. Seu objetivo é reduzir

o volume de dados na interação entre clientes e servidores, além de minimizar o custo de conexão na invocação de serviços e recuperação de dados, até mesmo quando falhas ocorrem.

Um exemplo de ação do DXS é ilustrado na figura 4.1, onde há um Serviço A com duas réplicas, Serviço A' e Serviço A", o que aumenta a disponibilidade do serviço durante invocações por clientes. Iniciamente, os parâmetros de requisição são enviados e armazenados para o DXS (1). Em seguida, uma ordem de invocação é definida pelo cliente na forma de um *workflow* (2). O processo de invocação (3) inicia-se pelo Serviço A e usa os serviços A' ou A" caso falhas ocorram. Em qualquer caso, os parâmetros são recuperados a partir do DXS (4) e os resultados parciais e final são armazenados também no DXS para recuperação posterior por outro serviço ou pelo cliente (5).

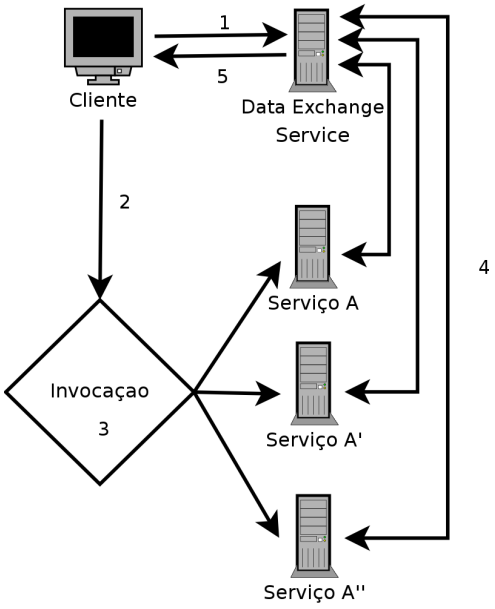

Figura 4.1: Uso do Data Exchange Service na Invocação de um Serviço Replicado [AD06]

O DXS trabalha de três formas diferentes. A primeira forma efetivamente estabelece *comunicação assíncrona* entre o cliente e o serviço alvo. O DXS intermedeia a comunicação e faz com que o cliente não necessite aguardar *online* por uma resposta do serviço alvo. Quando o processamento é concluído, o cliente recebe uma notificação do DXS, que informa que já é possível recuperar o resultado final.

O DXS também funciona como um *repositório temporário* para respostas intermediárias em uma cadeia de serviços. Resultados intermediários ou parciais são mantidos no DXS para o benefício de serviços ao longo da cadeia, mas ao cliente só é permitido acessar o resultado final quando este torna-se disponível.

Adicionalmente, o DXS suporta *tolerância a falhas*, uma vez que ele mantém informação sobre o estado da cadeia de serviços com seus respectivos resultados parciais. Com isso, recuperação e continuidade de processamento são possíveis pela reinvocação do serviço que falhou, utilizando os parâmetros armazenados. Assim, não se necessita reiniciar toda a cadeia de serviços, e são minimizados os reenvios de informação do cliente para os serviços-alvo. Isso é particularmente importante em serviços geográficos, nos quais o volume de dados que circula entre cliente e servidor pode ser expressivo.

Em todos os casos, o DXS funciona como um repositório de parâmetros e respostas de serviços e pode ser introduzido em qualquer *workflow*, o qual deve ser constituído apenas por serviços geográficos com interfaces em conformidade com o OGC.

Na figura 4.2 é apresentado um diagrama de seqüência em UML correspondente ao cenário de uso descrito na seção 3.1, onde se expõe a participação do DXS entre o cliente e os serviços geoespaciais de alto-nível usados.

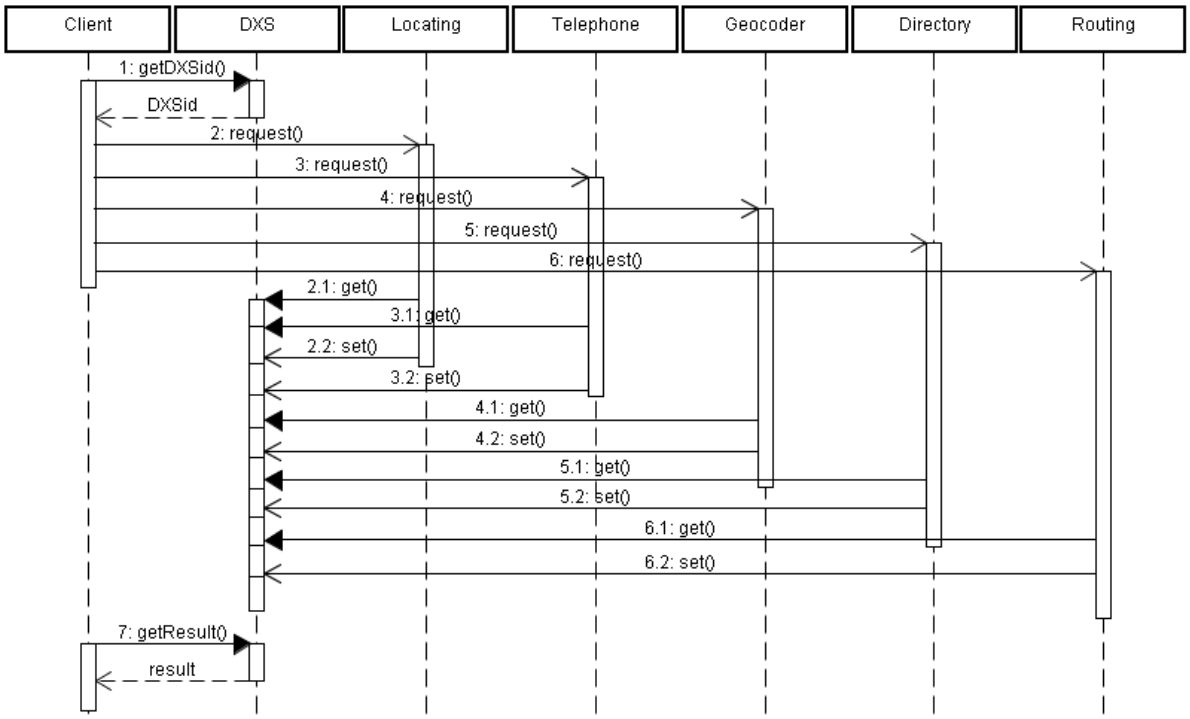

Figura 4.2: Diagrama de Seqüência em UML para um Sistema com DXS [AD06]

A mensagens enumeradas como 1 e 7, entre o cliente e o DXS, correspondem aos mesmos processos genéricos 1 e 5 ilustrados na figura 4.1, as quais iniciam o processamento e coletam o resultado final, respectivamente.

As mensagens 2, 3, 4, 5 e 6 invocam os serviços de interesse no início do processo, quando o cliente passa a aguardar pelo processamento enquanto os serviços mantêm-se

em execução. Como os serviços `Locating` e `Telephone` são executados em paralelo, as mensagens 2.1, 2.2, 3.1 e 3.2 podem ocorrer em qualquer ordem.

Finalmente, as mensagens de 4.1 a 6.2 ocorrem seqüencialmente. Nota-se que todo o tráfego intermediário passaria pelo cliente caso o DXS estivesse ausente. Entretanto, como o DXS assume o lugar do cliente neste intervalo, tal tráfego intermediário é retirado do cliente, o qual reassume sua função ao final. Além disso, algumas invocações na cadeia de serviços ocorrem em paralelo e reduzem o tempo de espera e reduzem o custo de controle para o cliente, uma vez que a responsabilidade pela recepção de respostas foi transferida para o DXS.

Portanto, em uma definição informal, o DXS tem a função de interceptar respostas de serviços e encaminhá-las para outros serviços (incluindo outros DXS) ou para o cliente. Neste sentido, os custos de comunicação para clientes gordos e ricos é o mesmo que sem o uso do DXS, enquanto que o custo para clientes magros é reduzido. Entretanto, o tamanho da *linha-da-vida* do cliente, assim como ilustrada na figura 4.2, permanece o mesmo, independentemente da presença ou não do DXS. Conseqüentemente, o DXS oferece vantagens somente quando o cliente tem limitações de energia, tem alto custo de comunicação ou é mais lento que o servidor do DXS.

É possível implementar persistência semelhante a suportada pelo DXS através do uso de serviços Web simples, mas uma interface padronizada, que trabalha como um serviço de infra-estrutura, é importante para assegurar a independência entre clientes e provedores de serviços OGC.

Por outro lado, o DXS pode apresentar alguns problemas de segurança, tais como acesso não autorizado a dados por terceiros. Melhorias no protocolo podem assegurar controle de acesso eficaz, quando somente serviços participantes de uma cadeia de serviços recuperam dados intermediários e nem mesmo ao cliente é permitido acessar dados privilegiados ou informação confidencial [BBC04].

4.2 Client Access Service

Como mencionado na seção anterior, o *Data Exchange Service* permite comunicação assíncrona entre o cliente e os serviços-alvo, ao permitir aos serviços o encaminhamento de seus resultados para o servidor do DXS, e ao cliente recuperar os resultados a partir do DXS em qualquer momento. Entretanto, invocações assíncronas não são suportadas em serviços Web da W3C por meio do protocolo HTTP [BCPR04, RLT05]. Entre os serviços OGC, somente o *Web Notification Service* implementa invocações assíncronas, apesar de este serviço não usar o protocolo HTTP [Ope].

Código 1 Uma Chamada que Retorna um identificador de Transação

```
transactionID =
  service.do(params);
```

Em um contexto urbano, serviços assíncronos são necessários quando há processos demorados em cadeias de serviço. Esse é frequentemente o caso de serviços baseados em um grande volume de dados geográficos. Eles são também necessários em aplicações específicas, tais como o uso de servidores para enviar dados a clientes através de comunicação não orientada a conexão.

Código 2 Obtenção do Resultado do Serviço

```
while (!service.isReady(transactionID))
  self.sleep(1000);
result = service.get(transactionID);
```

Apesar de o cliente usualmente não possuir um endereço IP (*Internet Protocol*) válido, ele sempre é capaz de acessar recursos por meio do protocolo HTTP. Entretanto, sem um endereço IP válido ele é inacessível a partir de outros clientes ou serviços, e recebe dados somente durante suas conexões de requisição. Uma alternativa é o cliente invocar um método de um serviço que retorna um ID de transação mas não retorna o resultado final, o que é ilustrado no código 1. Somente após algum período de tempo, uma nova invocação é realizada para obtenção do resultado final, quando o ID de transação é fornecido como parâmetro, assim como ilustrado no código 2.

Código 3 Cliente define um Método de Recepção de Notificação para quando o Resultado estiver Disponível

```
transactionID =
  service.do(responseMethod, responseURI,
             params)
```

Recuperar os resultados só é possível quando o processamento estiver concluído; porém, ao cliente não é dado conhecer o tempo que o processamento realmente consome. Dessa forma, o cliente consome recursos para monitorar o estado do processamento do serviço esperado. A figura 4.3 apresenta este padrão de comunicação.

Se o cliente tem um endereço IP válido e implementa funções de servidores HTTP, o cliente pode aguardar off-line pelo fim do processamento até o momento em que recebe uma notificação do serviço, informando sobre a disponibilidade do resultado. O código

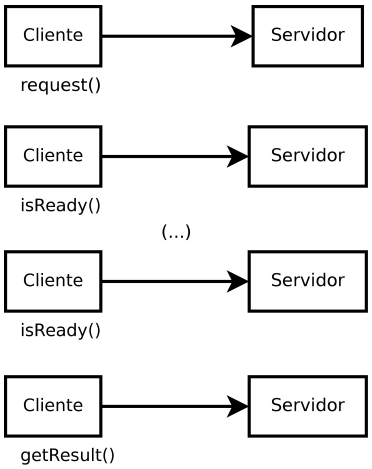

Figura 4.3: Cliente sem um Endereço IP Válido usando Comunicação baseada em HTTP [AD06]

3 ilustra uma chamada que especifica o método de resposta (HTTP, SMTP, SMS, etc) e uma URI como parte de seus parâmetros.

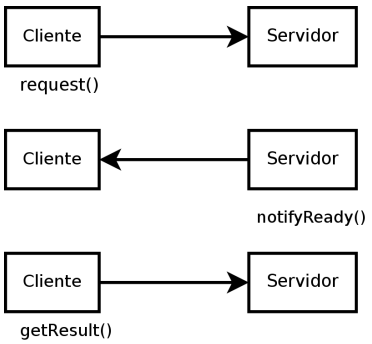

Figura 4.4: Cliente com um Endereço IP Válido usando Comunicação baseada em HTTP [AD06]

A figura 4.4 mostra o protocolo de comunicação descrito, enquanto o código 4 ilustra a última chamada do cliente ao serviço para obtenção do resultado final.

O último padrão discutido é dependente de um *gateway* local, tal como um serviço DXS, e pode ser adotado (1) quando o cliente não tem um endereço IP válido, (2) quando o cliente está protegido por um servidor *firewall* ou (3) quando espera-se que o cliente não seja identificado. Nestes casos, o cliente acessa o serviço alvo direta ou indiretamente, mas a notificação sobre a disponibilidade dos resultados ocorre entre o *gateway* escolhido e o cliente, como ilustrado na figura 4.5. O cliente implementa o código 3 e o código 4.

Ao fornecimento de funções de servidor HTTP (sem ou com IP válido) dá-se o nome de *Client Access Service* (CAS). O *Client Access Service* oferece meios de usar clientes

Código 4 Aguardar por Notificações do *Gateway* or outro Serviço

```
while (self.waiting(transactionID)) {
  // do nothing
}
result = service.get(transactionID);
```

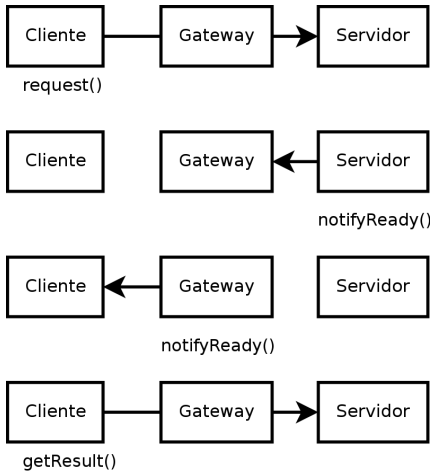

Figura 4.5: Cliente sem um Endereço IP Válido usando um *Gateway* como Mediador [AD06]

como sensores e como servidores para outros clientes, além de meios para habilitar o cliente a trabalhar como um provedor de *Data Exchange Service* (especialmente no caso de clientes gordos), o que é útil em aplicações nas quais o cliente envia resposta ao serviço a partir da experiência do usuário [Gio06], e particularmente útil nas situações em que um cliente envia dados para outros clientes próximos, sem que os dados passem pelo serviço de origem. Em todos os casos, o cliente pode prover alguma informação para outros clientes e serviços e suportar estas funcionalidades com adequado nível de segurança e privacidade.

4.3 Transaction Control Service

Através dos dois serviços apresentados anteriormente, pode-se melhorar a capacidade de processamento de um cliente de informação geográfica. Entretanto, estes serviços ainda não atendem ao desafio de desenvolver um cliente genérico, que funcione em diferentes LSDI. Além disso, a implementação de algumas características, tais como tolerância a falhas e transparência no desenvolvimento de clientes, são de responsabilidade do servidor ou de provedores específicos de informação geográfica. Transferir esta responsabilidade

para o cliente é conveniente em alguns casos, especialmente para clientes móveis em um contexto urbano.

Um cliente universal, apto a acessar transparentemente diferentes LSDI, deve ser independente de cadeias de serviço específicas. Neste caso, quando o cliente se encontra em um local distante ele pode carregar e executar uma cadeia de serviços diferente para obtenção de informações locais. Com isso, embora serviços possam estar prontamente disponíveis no interior de cadeias, seria melhor que cada serviço estivesse disponível individualmente para o cliente e fosse listado em um catálogo público. Dessa forma, a seleção de serviços e a formação de cadeias de serviços pode ocorrer dinamicamente sempre que o cliente necessite acessá-los.

O *Transaction Control Service* (TCS) executa transformações em um código que descreve uma cadeia de serviços genérica e em alto nível para produzir uma cadeia de serviços totalmente funcional, a qual serve para um contexto particular. O código genérico define um *workflow* básico que é seguido pelo cliente durante a execução de uma tarefa. Porém, apesar de tratar de *workflow*, o nome *transaction* (transação) foi adotado por este conjunto de funcionalidades. Isto se justifica pelo fato de o OGC definir uma transação como uma unidade lógica de trabalho constituída por uma ou mais operações de manipulação de dados, isto é, uma transação é um conjunto de uma ou mais invocações com leitura e escrita de dados, sem que estas operações atendam às propriedades de atomicidade, consistência, isolamento e persistência [Ope05b].

Através do TCS, o desenvolvedor de software-cliente define parâmetros que tornam possível a compilação do código de alto nível citado e o código executável específico para o cliente. Estes parâmetros são (1) a lista de serviços disponíveis, o que representa a LSDI escolhida, (2) o código que define a ordem de invocação, (3) as dependências das invocações, isto é, quais serviços são pré-requisito de outros e (4) o tipo de cliente para o qual o código executável será gerado.

O tipo de cliente, especificamente, serve como gabarito para os requisitos não-funcionais que devem ser atendidos para aquele cliente. No entanto, não se trata de um gabarito estático mas um conjunto flexível e personalizável de atributos, o que favorece a definição de tantos gabaritos quantos tipos de clientes existam, para que o workflow seja adaptado à disponibilidade de serviços na LSDI e às necessidades próprias de cada software-cliente.

Através dos códigos 5, 6 e 7 é apresentado o resultado do uso do TCS para a conversão do código abstrato (Código 5) em código genérico (Código 6), e finalmente em uma cadeia de serviços específica para um cliente magro, no cenário de assistência a viagem (Código 7). O principal objetivo deste serviço é portanto assegurar que nem o cliente nem o desenvolvimento do serviço sejam dependentes do contexto local, isto é, sejam independentes

de peculiaridades dos serviços que estiverem disponíveis em cada localidade, e que este arranjo possa ser refeito com flexibilidade caso mudanças ocorram no contexto.

O código 5 é apresentado como um pseudocódigo, apesar de ser especificado em XML. Nele estão presentes quatro blocos, sendo o primeiro e o terceiro com interação obrigatória entre a cadeia de serviços e o software-cliente, por necessidade de interação com o usuário. No segundo bloco, tornam-se explícitas duas condições, o paralelismo da obtenção da localização do cliente e da obtenção da lista de pontos de interesse próximos ao destino. Adicionalmente, a função `location(target)` mostra a dependência da função de obtenção de pontos de interesse com relação a esta função que retorna a localização do destino baseada em uma indicação dada pelo usuário (como por exemplo um endereço textual ou um número de telefone). No último bloco de código, a mesma função `location(target)` é usada, sendo que em sua invocação anterior não foi usado um identificador para "armazenar" o retorno (uma variável, em linguagem de programação). Dessa forma, ou haverá uma reinvocação do serviço ou pode ser usado o mesmo valor anterior, dependendo da semântica da função em questão, o que é definido por intervenção do programador.

Código 5 Código abstrato, independente de serviços e tecnologia

```
// Interacao com o usuario, controle com software-cliente
target = client.from();
myLocation = client.from();

// Comandos paralelos sem interacao com usuario
myLoc = location(myLocation,Tdirectory) |
 p = pointsOfInterest(location(target),Tdirectory);

//Interacoes com o cliente, controle com software-cliente
client.to(p);
p2 = client.from();

// Comandos nao interativos
query(Troute,myLoc, p2, location(target));
```

O código 6, por sua vez, apresenta mudanças no código anterior ao inserir o serviço *Data Exchange Service* em todas as invocações. Para obter a localização `myLoc` (do código anterior), é usada uma chave retornada pelo método `location`, a qual é armazenada no identificador com nome alterado `myLocAd`. Ao contrário deste primeiro método, o segundo `location` retorna a chave antes mesmo da invocação, a qual é encaminhada para o método `pointsOfInterest`, portanto a chave é criada pelo objeto `dxs` local através do método `dxs.setItem(0)`. O resultado da invocação do serviço correspondente ao `location` será

armazenado no servidor DXS, e o serviço de diretório obterá este resultado para usar como seu próprio parâmetro. Antes disso, o serviço de diretório retornará um endereço através do qual seu resultado poderá ser obtido pelo cliente, e este endereço será armazenado no identificador `pAd`. A interação com o cliente se mantém com os métodos antigos, mas só ocorre após o resultado do serviço de diretório estar disponível. Como no código anterior está definido que os pontos de interesse serão usados pelo cliente, o resultado do serviço é recuperado para o ambiente local e será manipulado diretamente pelo software-cliente. Finalmente, o ponto escolhido pelo usuário, através do software-cliente, é armazenado do identificador `p2` no servidor DXS, e o seu endereço no DXS em `p2Ad` será enviado para o serviço de roteamento, juntamente com os resultados da localização do cliente (`myLocAd`) e do destino (`location(target, dxs.setItem(0))`. É importante observar que se o destino for um objeto móvel e conseqüentemente houver necessidade de atualizar o resultado da invocação anterior, isto ocorrerá neste exemplo, pois há reinvocação do serviço de localização e o resultado é armazenado no mesmo espaço do DXS. Para que não ocorra reinvocação, o código precisaria ser alterado manualmente de `location(target, dxs.setItem(0))` para `dxs.getId(0)`.

O código específico deve implementar um comportamento de invocações que seja mais eficiente para o cliente [RCV06], o que deve ser projetado de acordo com as necessidades do cliente e particularidades da LSDI. Como exemplo, no trecho de código 7, é demonstrada a necessidade do cliente de obter a localização do destino pelo serviço mais rápido, sendo que qualquer dos serviços armazena a resposta no servidor DXS, em um mesmo espaço remoto no servidor DXS com endereço indexado no objeto `dxs` pelo método `setItem(0)`. Os serviços dos servidores da prefeitura (`cityHall`), da companhia telefônica (`teleCo`), da associação comercial local (`coAssociation`) e de um guia de turismo (`tourismGuide`) são invocados ao mesmo tempo, e a primeira resposta ficará disponível para o serviço invocado através do método `pointsOfInterest`, para o qual é informado o endereço do DXS retornado pelo método `dxs.getId(0)`.

Com os serviços apresentados, clientes de LSDI podem ser implementados de acordo com os requisitos informacionais e computacionais do OGC para serviços Web geoespaciais, reduzindo a introdução de aspectos tecnológicos no código do cliente. Assim, clientes gordos, ricos e magros podem ser implementados transparentemente, seus requisitos particulares são indicados por um *workflow* genérico que muda facilmente quando o cliente se move de uma localidade para outra e quando estes mesmos clientes têm sua capacidade alterada. Diferentes *workflows* são então especificados para cada infra-estrutura urbana por seus participantes e ainda para cada tipo de cliente por seu fabricante, o que habilita o software-cliente a adquirir comportamentos diferentes a partir do contexto.

Código 6 Código genérico adaptado para uma LSDI mas independente de tecnologia

```
// Inicializacao
loadServices(addr);
DXSService dxs = new DXSService();
dxs.start();

// Interacao com o usuario, controle com software-cliente
// Programador deve alterar os metodos de entrada de dados
target = client.from();
myLocation = client.from();

// Comandos paralelos sem interacao com usuario
myLocAd = location(myLocation,Tdirectory,dxs) |
 pAd =
 pointsOfInterest(location(target,dxs.setItem(0)),Tdirectory,dxs);

// Interacoes com o cliente, controle com software-cliente
dxs.waitItem(pAd);
client.to(dxs.getItem(pAd));
p2 = client.from();

p2Ad = dxs.store(p2);

// Comandos nao interativos
query(Troute, myLocAd, p2Ad, location(target,dxs.setItem(0)));
```

4.4 Implementação 3

Esta seção apresenta uma implementação voltada para a avaliação da facilidade com a qual são desenvolvidos clientes de LSDI que usam diretamente serviços Web geoespaciais propostos pela OGC. Porém, nesta implementação já são introduzidas funcionalidades dos serviços *Data Exchange Service* , *Client Access Service* e *Transaction Control Service* com o objetivo de sanar as limitações descritas na seção 3.4.

A Figura 4.6 ilustra o modelo implementado em apenas duas camadas, sendo a primeira do cliente e a segunda no servidor, onde localizam-se as fontes originais de informação e funcionalidades geográficas. O *Data Exchange Service* é adicionado em uma camada lógica, de onde assumirá algumas das funções do cliente durante o processamento dos serviços-alvo.

Nesta implementação, o cliente procura por novos serviços e os seleciona através de um catálogo de metadados universal. Portanto, o cliente e o servidor DXS conhecem a interface de cada serviço usado.

Código 7 Trecho de código específico adaptado para uma LSDI e para um cliente magro

```
// Comandos paralelos sem interacao com usuario
myLocAd = cityHall.location(myLocation, Tdirectory, dxs);

cityHall.location(target, dxs.setItem(0));
teleCo.location(target, dxs.setItem(0));
coAssociation.location(target, dxs.setItem(0));
tourismGuide.location(target, dxs.setItem(0));

pAd = cityHall.pointsOfInterest(dxs.getId(0), Tdirectory, dxs);
```

O DXS foi implementado como um serviço que recebe dados em GML e XML, retorna uma identificação, armazena os dados recebidos e os devolve caso solicitado em outra invocação. O CAS foi implementado como um padrão (*pattern*) diretamente no cliente, o qual responde por requisições HTTP e recebe a notificação de outros clientes ou servidores através de invocações a partir destes. O TCS, por sua vez, foi prototipado como um documento XSLT que faz sucessivas transformações em um documento XML e devolve métodos Java prontos para a implementação de clientes que usam os serviços Web geoespaciais de interesse.

4.5 Avaliação

No capítulo 3 foram avaliadas as especificações do modelo abstrato da OGC para serviços Web geoespaciais dos conjuntos OpenLS e OWS, e, a partir das limitações encontradas e apresentadas na seção 3.4, foram desenvolvidos os serviços especificados no presente capítulo.

A avaliação inicial foi desenvolvida sobre a implementação de dois modelos que são (a) 1 Cliente para 1 Provedor e n Servidores de serviços OGC, onde há um provedor que intermedeia recursos de outros provedores, estando estes últimos em conformidade com os serviços OGC (Figura 3.3), e (b) 1 Cliente para n Servidores, onde o cliente executa todas as tarefas relacionadas ao acesso a serviços Web e processamento da informação geográfica (Figura 3.4).

Após as duas primeiras implementações, foram separados os problemas de projeto dos problemas tecnológicos encontrados. Os problemas de projeto foram então agrupados em três conjuntos, que constituem os serviços *Data Exchange Service* , *Client Access Service* e *Transaction Control Service* . Finalmente, a avaliação foi baseada no modelo (b) de duas camadas com a inclusão do DXS, o que foi especificado e verificado na seção 4.4.

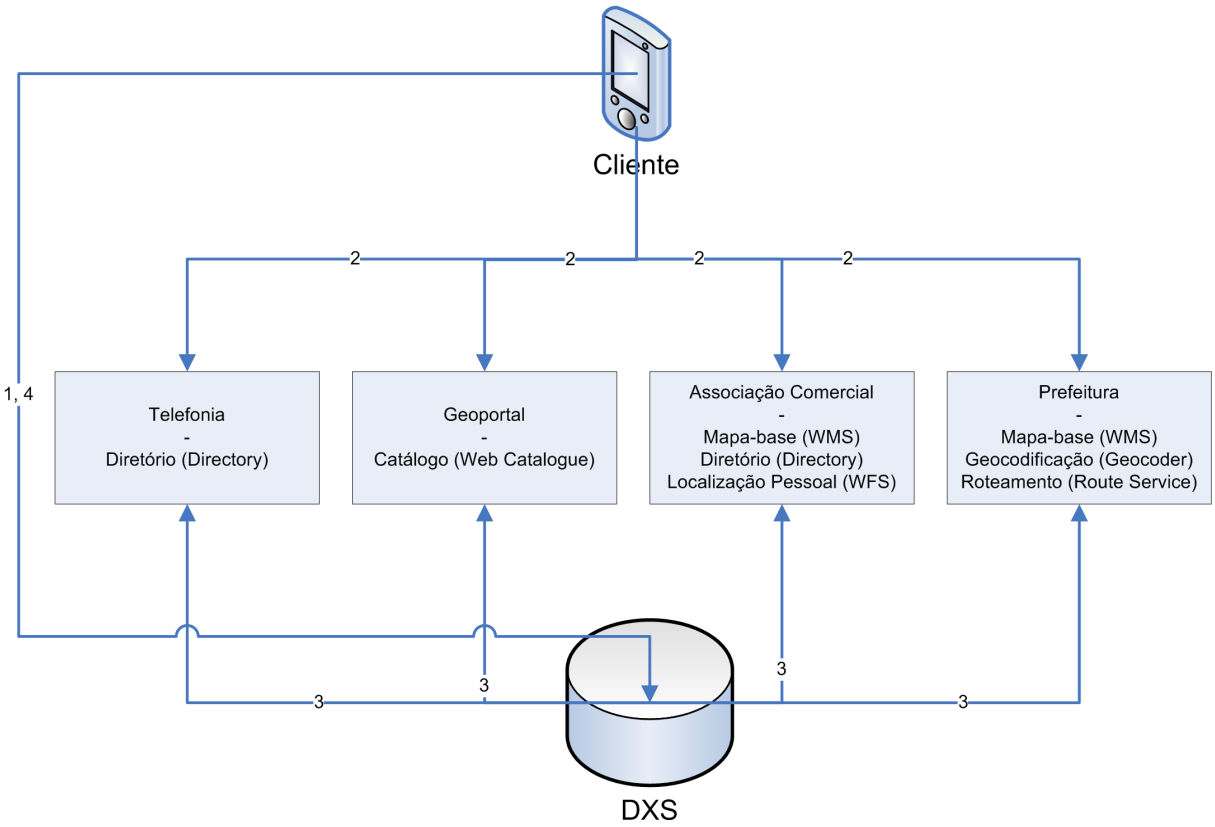

Figura 4.6: Implementação 3, onde o cliente acessa diretamente os serviços-alvo mas adota um DXS para armazenamento temporário

Pela implementação dos recursos reunidos no *Data Exchange Service* , o tráfego intermediário de dados foi eliminado do cliente durante o processamento independente de interações com o usuário. Adicionalmente, quando introduzidas falhas por meio de intervenções no ambiente de execução, nenhum dado previamente processado foi perdido.

Entretanto, não foram observados efeitos para os RNF economia de comunicação, economia de processamento, economia de espaço e velocidade de acesso durante o período de processamento no qual havia interação com o usuário, quando há novos parâmetros para a cadeia de serviço advindos diretamente do cliente.

Através do *Client Access Service*, os RNF economia de comunicação e velocidade de acesso foram melhorados pela eliminação do uso da rede de computadores pelo cliente durante o período de processamento do serviço remoto. Isso tornou-se possível pelo fato de que o cliente passou a ser informado sobre o estado de prontidão do serviço.

O CAS emulou comunicação assíncrona através da atribuição da função de servidor HTTP ao cliente, o que se mostrou inviável em certas condições (como no caso em que o cliente não possui um endereço IP válido). O mesmo comportamento tornou-se possível

pelo uso de um protocolo diferente do HTTP para o envio de uma mensagem ao cliente, após a qual o cliente reinvoca o servidor através do protocolo HTTP para obter o resultado final.

Finalmente, através do *Transaction Control Service* foi possível a compilação de métodos específicos para dispositivos e clientes, de forma transparente, a partir de uma definição de alto-nível em XML. Esse serviço pôde funcionar como um compilador que transforma um documento XML em métodos Java de acordo com o *workflow* definido pelo desenvolvedor, dado um conjunto de requisitos do cliente.

Os métodos Java resultantes incluem o uso do servidor DXS quando este se faz necessário, como por exemplo na implementação de código para clientes magros. Assim, foi possível a construção de código de alto-nível para clientes gordos, ricos e magros transparentemente, o qual pode ser transformado em código específico para dispositivos em atendimento aos requisitos de cada tipo de dispositivo.

Capítulo 5

Conclusões

Assim como definido no Capítulo 1, o principal objetivo desta dissertação foi avaliar a adequação de serviços assim como especificados pela OGC nos conjuntos OWS e OpenLS para a implementação de sistemas de informação geográficos de âmbito urbano e implementação de clientes para estes mesmos sistemas. No entanto, o foco foi dado principalmente a requisitos não funcionais de engenharia de sofware, como uma forma de investigar a facilidade de desenvolvimento e, conseqüentemente, a contribuição desses padrões para a difusão de aplicações geográficas de escopo urbano.

Apesar de SDI possuirem vários níveis de detalhamento e complexidade, que vão desde SDI corporativas até a SDI global, a padronização de interfaces de serviços foi realizada tendo em vista principalmente SDI nacionais e regionais. Com isso, esperava-se fomentar o desenvolvimento da SDI global, como uma única infra-estrutura que permitisse que os SIG de todo o mundo cooperassem entre si, assim como ocorre com a Internet em relação a redes de computadores menores. Inevitavelmente, SDI locais e corporativas não implementaram os padrões da OGC nos últimos anos, e em alguns casos, sequer os protocolos da W3C foram adotados. Com isso, inúmeras SDI se desenvolveram como "ilhas", atendendo apenas aos requisitos de seus usuários locais, o que dificulta a integração destas ilhas entre si, e a integração destas com níveis acima, como com a SDI nacional na qual deveriam estar inseridas.

O que não representava um problema há dez anos atrás, atualmente, com o advento e difusão de softwares para computadores pessoais (por exemplo, o *Google Earth*), que reúne uma grande comunidade de usuários que colecionam pontos de interesse (normalmente urbanos), e outros softwares de localização para veículos, PDAs e celulares, a importância das LSDI foi ampliada. Isto deve-se principalmente ao fato de que as LSDI devem servir como fonte de informação detalhada sobre as diversas regiões do planeta, o que só é possível quando estas se integram às NSDI e à GSDI.

Os resultados da pesquisa, apresentados na seção 5.1, referem-se a aspectos relacionados à classificação das limitações encontradas, à modelagem de cenário de uso adotado para avaliação e ao protótipo desenvolvido como prova de conceito.

As contribuições da dissertação são descritas em seguida, na seção 5.2, onde são enumerados os três serviços desenvolvidos, os quais agrupam as técnicas de satisfação usadas para reduzir as limitações identificadas e para melhorar a facilidade de desenvolvimento de SIG urbanos.

Finalmente, na seção 5.3 são apontadas algumas indicações da direção para a qual novos estudos podem ser conduzidos, os quais devem contribuir principalmente para reduzir as diferenças entre os padrões de serviços Web geoespaciais da OGC e os de serviços Web genéricos da W3C.

5.1 Resultados

Neste trabalho, foram avaliados aspectos de engenharia das especificações *OGC Web Services* e *Open Location Services*, ambas definidas pela OGC, tendo sido considerada também a especificação de *Web services* definida pela W3C, todas aplicadas a sistemas de informação geográficos de escopo urbano. Por meio da implementação de um protótipo composto por serviços geográficos e clientes foram identificadas limitações para o desenvolvimento de SIG distribuídos, tais como ausência de mecanismos de tolerância a falhas padronizados e clientes fortemente dependentes de provedores, dentre outras. Uma vez que a disponibilidade de softwares-cliente contribui para a difusão de LSDI, a facilidade com a qual estes clientes são desenvolvidos é muito importante.

Algumas soluções para as limitações encontradas foram agrupadas em novos serviços de infra-estrutura, os quais facilitam o desenvolvimento de clientes universais não dependentes de fornecedores específicos para LSDI [AD06]. Entretanto, tais serviços não constituem propostas para a OGC, eles apenas sintetizam e demonstram as limitações encontradas.

A modelagem do cenário de uso real foi mapeada em serviços de perspectiva computacional (WMS, WFS, WCS, etc) após ter sido especificada como um conjunto de serviços de alto nível na perspectiva de negócio (mapa-base, roteamento, diretório de empresas, etc). Com isso, esta pesquisa experimentou a modelagem de LSDI proposta por Davis Jr e Alves [DA06].

O protótipo, por sua vez, se baseou no modelo abstrato de serviços da OGC, o que adicionalmente possibilita usá-lo como instrumento de análise para outras questões relacionadas a OGC em esforços futuros de pesquisa. Além dos serviços desenhados, esta

pesquisa organizou alguns dos requisitos não funcionais dos principais serviços da OWS e OpenLS o que poderá ser estendido em próximos trabalhos.

De fato, as especificações da OGC mostraram-se inadequadas para a implementação de SIG urbanos assim como desenvolvidos de outras formas, como pela adoção de pacotes proprietários de desenvolvimento e pela imposição de interfaces não-padronizadas. Um passo na direção das LSDI deve ser dado pela OGC, para garantir uma adoção ampla de seus padrões e fomentar a consolidação da SDI global [Rei05].

5.2 Principais Contribuições

Os serviços da LSDI e o SIG específico foram construídos iterativamente sobre o cenário de uso especificado na seção 3.1 e requisitos não funcionais definidos no ORM [Per03] na perspectiva de Engenharia. Durante a implementação, para cada regra de correlação que influenciava negativamente algum dos requisitos não funcionais, uma técnica de satisfação era aplicada, por meio da reimplementação do serviço em conformidade com a OGC, da implementação de um outro serviço OGC, ou pela implementação de funcionalidades adicionais, em um ou mais serviços, que pudessem reduzir o impacto negativo percebido.

Essas funcionalidades adicionais foram organizadas em grupos segundo os requisitos não funcionais para os quais eram destinadas. Os três grupos foram então definidos como *Data Exchange Service* (DXS), *Client Access Service* (CAS) e *Transaction Control Service* (TCS).

O Serviço de Intercâmbio de Dados (DXS) provê a função de persistência, envolvendo provedores de serviço e o espaço de armazenamento local do cliente. Com o DXS se tem um serviço neutro, através do qual a cadeia de serviços troca parâmetros e resultados de seu processamento interno, sem uso do canal de comunicação do cliente e sem atribuição da responsabilidade pela recepção de resultados intermediários ao cliente.

As cadeias de serviços geográficos, atualmente, são construídas em um serviço remoto, seja ele geográfico (OGC) ou um serviço genérico (baseado nas especificações W3C). Estas cadeias já se beneficiam da capacidade de comunicação entre servidores. Entretanto, cadeias de serviços da W3C muitas vezes são orquestradas por clientes, que são integralmente responsáveis pelas invocações, recepção de resultados intermediários e encaminhamento destes resultados aos próximos serviços da cadeia. Essa segunda condição não necessariamente exige que o dispositivo onde reside o software cliente tenha capacidade similar a de servidores, mas exige no mínimo que não existam certas restrições, tais como custo elevado de comunicação e baixa capacidade de energia, comuns a dispositivos para os quais SIG são ferramentas úteis (celulares, GPS, PDAs, e outros).

Em ambos os casos, o software cliente (e o usuário, conseqüentemente) tem alguma capacidade de escolha. Se cadeias de serviços são construídas diretamente em um provedor, o cliente pode trocar os serviços internos à cadeia pela escolha de outro provedor ou outro serviço do mesmo provedor, isto é, troca-se toda a cadeia de serviços. Porém, se a cadeia de serviço é construída dinamicamente pelo cliente, tem-se maior flexibilidade, pois serviços constituintes da cadeia podem ser substituídos por outros serviços compatíveis com facilidade, inclusive em tempo de execução, o que é útil quando o cliente muda de região ou quando os parâmetros de segurança, qualidade e custo são alterados. Além disso, se a cadeia é responsabilidade do cliente, outros requisitos tais como tolerância a falhas também são transferidos para o cliente. Isso traz vantagens e desvantagens, pois adquire-se alguma independência de provedores, mas exige-se a inclusão de código que não pertence ao domínio da aplicação.

Mesmo assim, o DXS auxilia na transferência da responsabilidade pela orquestração da cadeia para o cliente ao assumir a função de atender os requisitos não funcionais. Dessa forma ele permite reduzir o reenvio de parâmetros iniciais em caso de falhas da cadeia de serviços e permite o aproveitamento de resultados intermediários quando apenas algum componente da cadeia falha, o que é útil até mesmo para clientes gordos. Porém, as principais vantagens são para clientes magros e ricos, os quais normalmente apresentam restrições mais sérias de energia, armazenamento, comunicação e capacidade de processamento.

Por meio do serviço de acesso ao cliente (CAS) foi possível emular assincronicidade em invocações usando o protocolo de aplicação HTTP. Este serviço foi incorporado ao cliente, o qual passou a se comportar como um servidor HTTP, com função de responder por requisições de qualquer outro cliente ou servidor. Diferentemente do serviço de intercâmbio de dados, o serviço de acesso ao cliente não é um serviço Web. Portanto, sua implementação constituiu um comportamento introduzido no cliente e avaliado para o suporte a transações longas, muito comuns no contexto de SIG, onde há volume maior de dados e dificuldades particulares de indexação.

Outro conjunto de funções avaliado pertence ao grupo do serviço de controle de transações (TCS), constituído basicamente por transformadores de código de alto nível para código de programação. Através deste serviço foi possível experimentar a introdução das novas funcionalidades de forma automática no cliente, possibilitando a implementação transparente de clientes para diferentes dispositivos como gordos, ricos ou magros. A importância deste serviço está associada a capacidade de implementação de clientes adequados para SIG urbanos, todos porém em conformidade com os padrões da OGC, o que favorece a consolidação de uma infra-estrutura global de dados espaciais, dentre outras

coisas [Rei05].

Portanto, esta dissertação propôs um conjunto de novos serviços, os quais facilitam o desenvolvimento de clientes de informação geográfica para sistemas baseados em LSDI, e conseqüentemente contribuem para melhorar a implementação e difusão de LSDI como fontes de informação detalhada para SDI nacionais, regionais e global.

5.3 Trabalhos Futuros

Foram identificadas três direções principais para trabalho futuro. Primeiramente discute-se o uso potencial de clientes como provedores de informação geográfica. Em seguida, propõem-se melhoramentos de engenharia para o desenvolvimento de serviços geográficos. Finalmente, supõe-se a possibilidade de desenvolvimento de novos serviços de alto nível para LSDI, melhorando a técnica de modelagem, e desenvolvimento de um novo protótipo baseado no modelo de implementação da OGC para favorecer outros estudos nas perspectivas computacional e tecnológica.

O cliente de informação geográfica, se visto como um servidor de parâmetros para a cadeia de serviços, torna-se potencialmente o provedor de vários tipos de informação. Informações coletadas diretamente pelo usuário, como estado de tráfego [TJH05] juntamente com dados previamente coletados de outras fontes, qualidade percebida de informação, ontologias sobre seus interesses, observações ambientais [Gio06], posição geográfica atualizada e outras, todas podem ser transmitidas para outros usuários ou melhorar a qualidade da informação fornecida por um provedor de serviços. Dessa maneira, algumas aplicações tornam-se possíveis, como planejamento de rotas mais preciso pela informação trocada com outros usuários, controle de qualidade de serviço em provedores, uma melhor gerência de tráfego por meio de veículos que servem como sensores em tempo-real, dentre outras.

Aplicações geográficas através de dispositivos celulares, PDAs e outros podem ser executadas através de comunicação cliente-servidor ou conexões *peer-to-peer*. No caso de conexões *peer-to-peer*, quando a disponibilidade de energia não for limitada, pode-se melhorar o tempo de resposta das aplicações [WZK05] uma vez que dados anteriormente coletados por outros usuários estarão imediatamente disponíveis [GWZ04]. A comunicação *peer-to-peer* aproxima-se muito do comportamento do *Client Access Service* e serve para implementar este padrão. Serviços de geocodificação, roteamento e serviços de localização podem ainda ser adaptados para objetos geográficos móveis, tais como veículos (de transporte público, de emergência ou particulares) ou pessoas (por exemplo, por meio de telefones celulares ou GPS). No entanto, estudos adicionais sobre tais aplicações são requeridos por envolverem questões relacionadas a segurança e privacidade.

Uma questão de pesquisa relacionada a engenharia de SIG envolve determinar como parâmetros de tolerância a falhas podem ser definidos e negociados através de contratos de disponibilidade, os quais podem ser automaticamente verificados por meio de buscas no interior das cadeias de serviço, quando são identificados pontos de falha comuns entre um serviço e seus substitutos. Isso é especialmente importante em emergências, aplicações de missão crítica e outras [Onc05]. Esta mesma questão pode ser estendida para outros aspectos, tais como privacidade e desempenho.

Uma segunda questão em desenvolvimento de SIG refere-se às descrições de cadeias de serviço ou *workflow*. Em alguns casos, é útil que serviços sejam localizados e dinamicamente carregados no interior de uma cadeia de serviços, como no caso de serviços de roteamento quando rotas passam por regiões não cobertas por fontes de dados selecionadas. Portanto, melhorias adicionais devem ser feitas nos mecanismos de descrição de cadeias de serviço e em mecanismos que permitam transformar tais descrições em código.

Novos meios de avaliação de LSDI são também essenciais para o desenvolvimento de SIG urbanos como componentes das NSDI. Davis Jr e Alves (2005) [DA05] propõem vários serviços de alto nível que constituem e permitem modelar uma LSDI ao considerar necessidades específicas para aplicações geográficas urbanas. O arcabouço pode ser estendido pela proposição de novos serviços relacionados a esse nível de infra-estrutura de dados especiais.

Finalmente, é importante buscar a criação de um ambiente de desenvolvimento de protótipos e avaliação de LSDI, pois tal ambiente pode possibilitar observações e experimentos em padrões da OGC atuais e futuros, assim como possibilitar estudos adicionais sobre vários aspectos relacionados a problemas reais que limitam a adoção em larga escala de LSDI compatíveis, tais como privacidade dos usuários de SIG, desempenho das aplicações, segurança, disponibilidade e outros.

Referências

[AB03] Leonardo Lacerda Alves and Fabricio Roulin Bittencout. PHP: Conceitos essenciais para implementação de aplicações web. *7 Faces*, 4(1):193–208, 2003.

[AD06] Leonardo Lacerda Alves and Clodoveu Augusto Davis Júnior. Interoperability through Web Services: Evaluating OGC Standards in Client Development for Spatial Data Infrastructures. In *VIII Brazilian Symposium on Geoinformatics Proceedings*, pages 193–208. INPE, November 2006.

[AD07] Leonardo Lacerda Alves and Clodoveu Augusto Davis Júnior. *Evaluation of OGC Web Services for Local Spatial Data Infrastructures and for Development of Geographic Information Systems Clients*, volume 1. Springer, S.L., 2007.

[AEMS05] D. Askew, S. Evans, R. Matthews, and P. Swanton. Magic: a geoportal for the english countryside. *Computers, Environment and Urban Systems*, 29(1):71–85, 2005.

[Ala03] Nadine Alameh. Chaining geographic information web services. *IEEE Internet Computing*, 7(5):22–29, 2003.

[BBC04] A. Belussi, E. Bertino, and B. Catania. An authorization model for geographical maps. In *GIS'04 Proceedings*, pages 82–91. ACM, November 2004.

[BC05] Lars Bernard and Max Craglia. SDI – From Spatial Data Infrastructure to Service Driven Infrastructure. In *Research Workshop on Cross-learning between Spatial Data Infrastructures and Information Infrastructures*, 2005.

[BCPR04] M. Brambilla, S. Ceri, M. Passamani, and A. Riccio. Managing asynchronous web services interactions. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 80–87. IEEE, November 2004.

[BEK+00] Ian D. Bishop, Francisco J. Escobar, Sadasivam Karuppannan, Ksemsan Suwarnarat, Ian P. Williamson, Paul M. Yates, and Haider W. Yaqub. Spatial data infrastructures for cities in developing countries: Lessons from the Bangkok experience. *Cities*, 17(2):85–96, April 2000.

[BLM05] P. Beaumont, P. A. Longley, and D. J. Maguire. Geographic information portals: a uk perspective. *Computers, Environment and Urban Systems*, 29(1):49–69, 2005.

[Boe98] B. Boehm. Using the winwin spiral model: A case study. *Computer*, 31(7):33–44, July 1998.

[CBRW04] J. Crompvoets, A. Bregt, A. Rajabifard, and I. Williamson. Assessing the worldwide developments of national spatial data clearinghouses. *International Journal of Geographical Information Science*, 18(7):665–689, 2004.

[CCD+05] Marco Antônio Casanova, Gilberto Câmara, Clodoveu A. Davis Junior, Lubia Vinhas, and Gilberto Ribeiro de Queiroz. *Bancos de Dados Geográficos*, volume 1. EspaçoGeo, Curitiba, 2005.

[CFRW01] Tai On Chan, Mary-Ellen Feeney, Abbas Rajabifard, and Ian Williamson. The dynamic nature of spatial data infrastructures: A method of descriptive classification. *Geomatica*, 55(1):65–72, 2001.

[CPD06] S. Chung, J. R. Pan, and S. Davalos. A special web service mechanism: Asynchronous .NET web services. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 212–212. IEEE, February 2006.

[DA05] Clodoveu Augusto Davis Júnior and Leonardo Lacerda Alves. Local spatial data infrastructures based on a service-oriented architecture. In *VII Brazilian Symposium on Geoinformatics Proceedings*, pages 84–89. INPE, November 2005.

[DA06] Clodoveu Augusto Davis Júnior and Leonardo Lacerda Alves. Infraestruturas de dados espaciais: Potencial para uso local. *IP. Informática Pública*, 8(1), Março 2006.

[DA07] Clodoveu Augusto Davis Júnior and Leonardo Lacerda Alves. *GeoSpatial Web Services. Book chapter in: Encyclopedia of Geographic Information Systems, to appear*. Springer-Verlag, 1st edition, 2007.

[Dee06] Deegree. *Deegree Project (WMS, WFS, WCS, iGeoPortal, deeJUMP)*. Lat/-Lon GmbH http://www.deegree.org, 2006.

[Dem06] Demis. *Demis Web Map Server*. Demis http://www.demis.nl/, 2006.

[DF06] Clodoveu A. Davis Júnior and Frederico Fonseca. Considerations from the development of a local spatial data infrastructure. *Information Technology for Development*, 12(4):273–290, 2006.

[DL99] Clodoveu Augusto Davis Junior and Alberto Henrique Frade Laender. Multiple representations in GIS: materialization through map generalization, geometric, and spatial analysis operations. In *ACM GIS'99: Proceedings of 7th International Symposium on Advances in Geographic Information Systems*, pages 60–65, New York, NY, USA, 1999. ACM Press.

[FCOM06] Frederico Fonseca, Gilberto Câmara, H Onsrud, and A M Monteiro. Networks of Innovation and the Establishment of a Spatial Data Infrastructure in Brazil. *Information Technology for Development*, 12(4):255–272, 2006.

[FGD97] FGDC Federal Geographic Data Committee. Metadata to Clearinghouse Hands-On Tutorial, Washington, DC, 1997.

[FM05] Frederico T. Fonseca and James E. Martin. Toward an alternative notion of information systems ontologies: Information engineering as a hermeneutic enterprise. *Journal of the American Society for Information Science and Technology*, 56(1):46–57, 2005.

[GGR05] Carlos Granell, Michael Gould, and Francisco Ramos. Service composition for SDIs: Integrated components creation. In *Proceedings of II Internation on Geographic Information Management*, August 2005.

[Gho01] Rina Ghose. Use of information technology for community empowerment: transforming geographic information systems into community information systems. *Transactions in GIS*, 2(5):141–163, January 2001.

[Gio06] Francisco Luiz Pompéia Gioielli. Tecnologias e padrões abertos para o domínio geográfico na web: um estudo em ecoturismo. Master's thesis, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, INPE-13779-TDI/1053, 2006.

[GWZ04] Jihong Guan, Leichun Wang, and Shuigeng Zhou. Enabling GIS services in a P2P environment. In *Computer and Information Technology, 2004. CIT '04. The Fourth International Conference on*, pages 776 – 781. IEEE, September 2004.

[HC01a] Cay S. Horstmann and Gary Cornell. *Core Java 2: Volume I - Fundamentos*. Makron Books, 1st edition, 2001.

[HC01b] Cay S. Horstmann and Gary Cornell. *Core Java 2: Volume II - Advanced Features*. Prentice Hall PTR, 5th edition, 2001.

[HS05] Michael N. Huhns and Munindar P. Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, 9(1):75–81, 2005.

[INS02] INSPIRE Architecture and Standards Working Group. INSPIRE Architecture and Standards Position Paper, Brussels, Comission of the European Community, 2002.

[JSTW02] Steve Jacoby, Jessica Smith, Lisa Ting, and Ian Williamson. Developing a common spatial data infrastructure between state and local government: An australian case study. *International Journal of Geographical Information Science*, 16(4):305–322, 2002.

[KSR05] P. Kumar, V. Singh, and D. Reddy. Advanced traveler information system for Hyderabad City. *Intelligent Transportation Systems, IEEE Transactions on*, 6(1):26–37, March 2005.

[LCdQ01] P. Lima, Gilberto Câmara, and Gilberto R. de Queiroz. Intercâmbio de Dados Geográficos: Modelos, Formatos e Conversores. In *III Workshop Brasileiro de Geoinformática*, November 2001.

[LCdQ02] P. Lima, Gilberto Câmara, and Gilberto R. de Queiroz. GeoBR: Intercâmbio Sintático e Semântico de Dados Espaciais. In *IV Brazilian Symposium on Geoinformatics*, November 2002.

[LPD05] J. Lieberman, T. Pehle, and M. Dean. Semantic Evolution of Geospatial Web Services. In *W3C Workshop on Frameworks for Semantic Web Services*, 2005.

[Mab04] Marwa Mabrouk. *OpenGIS Location Services (OpenLS): Core Services, OGC document 03-006r3.* OGC, Wayland, 2004.

[Man06] W. H. Erik De Man. Understanding SDI; complexity and institutionalization. *International Journal of Geographical Information Science*, 20(3):329–343, March 2006.

[Mas99] Ian Masser. All shapes and sizes: The first generation of national spatial data infrastructures. *International Journal of Geographical Information Science*, 13(1):67–84, 1999.

[Mas05] Ian Masser. Some priorities for SDI related research. In *Proceedings of the FIG Working Week and GSDI 8: From Pharaohs to Geinformatics, Cairo, Egypt*, page 11. FIG, April 2005.

[MCN92] John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and using non-functional requirements: a process-oriented approach. *Software Engineering*, 18(6):483–497, 1992.

[ML05] David J. Maguire and Paul A. Longley. The emergence of geoportals and their role in spatial data infrastructures. *Computers, Environment and Urban Systems*, 29(1):3–14, 2005.

[MRZW06] A. Mansourian, A. Rajabifard, M. J. Valadan Zoej, and I. Williamson. Using SDI and web-based system to facilitate disaster management. *Computers & Geosciences*, 32(3):303–315, April 2006.

[NBFRW04] Zorica Nedovic-Budic, Mary-Ellen F. Feeney, Abbas Rajabifard, and Ian Williamson. Are SDIs serving the needs of local planning? case study of Victoria, Australia and Illinois, USA. *Computers, Environment and Urban Systems*, 28(4):329–351, July 2004.

[Onc05] Richard Onchaga. Towards Quality-aware Composition of Geo-services. In *Proceedings of the FIG Working Week and GSDI 8: From Pharaohs to Geinformatics*, page 11. GSDI Association, April 2005.

[Ope] Open Geospatial Consortium. *Sensor Web Enablement Architecture, OGC document 05-090.* OGC, Wayland.

[Ope05a] Open Geospatial Consortium. *OGC Catalogue Services Specification, OGC document 04-021r3.* OGC, USA, 2005.

[Ope05b] Open Geospatial Consortium. *Web Feature Service Implementation Specification, OGC document 04-094.* OGC, USA, 2005.

[Per03] George Percivall. OGC Reference Model OGC 03-040. *Open Geospatial Consortium, Inc*, 2003.

[Pre05] Roger S. Pressman. *Engenharia de Software.* McGraw Hill, 5th edition, 2005.

[PWE99] Andrew Phillips, Ian Williamson, and Chukwudozie Ezigbalike. Spatial data infrastructure concepts. *The Australian Surveyor*, 44(1):20–28, 1999.

[RCV06] Jose Geraldo Ribeiro Junior, Glauber Tadeu S. Carmo, and Marco Tulio O. Valente. Invocation of replicated web services using smart proxies. In *WebMedia '06: Proceedings of the 12th Brazilian symposium on Multimedia and the web*, pages 138–147, New York, NY, USA, 2006. ACM Press.

[Ref06] Refractions Research. *OGC Survey.* Refractions http : // www.refractions.net / white_papers / ogcsurvey, Victoria, Canada, 2006.

[Rei05] Mark Reichardt. GSDI Depends on Widespread Adoption of OGC Standards. In *Proceedings of the FIG Working Week and GSDI 8: From Pharaohs to Geinformatics*, page 12. GSDI Association, April 2005.

[RLT05] M. Ruth, Feng Lin, and Shengru Tu. A client-side framework enabling callbacks from web services. In *Web Services, 2005. ECOWS 2005. Third IEEE European Conference on*, page 12. IEEE, November 2005.

[RWHJ00] Abbas Rajabifard, Ian P. Williamson, Peter Holland, and Glenn Johnstone. From local to global SDI initiatives: a pyramid of building blocks. In *IV Global Spatial Data Infrastructure Conference Proceedings*, March 2000.

[SDB+05] L. A. Souza, C. A. Davis Júnior, K. A. V. Borges, T. M. Delboni, and A. H. F. Laender. The Role of Gazetteers in Geographic Knowledge Discovery on the Web. In *(LA Web 2005): Proceedings of the 3rd Latin American Web Congress*, 2005.

[SKK06] Marius Scholten, Ralf Klamma, and Christian Kiehle. Evaluating performance in spatial data infrastructures for geoprocessing. *IEEE Internet Computing*, 10(5):34–41, 2006.

[Sky06] Skylab. *J2ME OGC WMS Client*. Skylab http :// www . skylab-mobilesystems . com / en / products / j2me_wms_client . html, 2006.

[Son04] J. Sonnet. *OWS 2 Common Architecture: WSDL, SOAP e UDDI*. OGC, Wayland, 2004.

[TA90] B. H. Tay and A. L. Ananda. A survey of remote procedure calls. *ACM Operating Systems Review*, 24(3):68–79, July 1990.

[Tai05] M. G. Tait. Implementing geoportals: Applications of distributed gis. *Computers, Environment and Urban Systems*, 29(1):33–47, 2005.

[TAM03] Jan Turkstra, Nelly Amemiya, and Jose Murgia. Local spatial data infrastructure, Trujillo-Peru. *Habitat International*, 27(1):669–682, 2003.

[TJH05] Xiaofeng Tao, Changjun Jiang, and Yaojun Han. Applying SOA to intelligent transportation system. In *Proceedings of the 2005 IEEE International Conference on Services Computing*, pages 101–104. IEEE, July 2005.

[UCF+05] Helton Nogueira Uchoa, Renata Juliana Cristal Coutinho, Paulo Roberto Ferreira, Luiz Carlos Teixeira Coelho Filho, and Jorge Luís Nunes e Silva Brito. Arquitetura OpenGis Baseada em Software Livre para Solução de Geoprocessamento. In *Anais do XXII Congresso Brasileiro de Cartografia*, 2005.

[Uni98] United States Geological Survey. Spatial Data Transfer Standard, 1998.

[Whi05] Arliss Whiteside. *OpenGIS Web Services Common Specification, OGC document 05-008*. OGC, Wayland, 2005.

[Wor04] World Wide Web Consortium. *Web Services Architecture W3C Working Group Note (Feb. 11 2004)*. W3C. www.w3.org/TR/2004/NOTE-ws-arch-20040211/, 2004.

[WZK05] Haojun Wang, Roger Zimmermann, and Wei-Shinn Ku. ASPEN: An adaptive spatial peer-to-peer network. In *GIS'05 Proceedings*, pages 230–239. ACM, November 2005.

Apêndice A

Transaction Control Service

A.1 Entrada

O documento XML seguinte apresenta o pseudocódigo apresentado no capítulo 4 em seu formato de entrada original.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="java.xslt"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="tcs.xsd">
 <comm>
  <type>input</type>
  <value>target</value>
 </comm>
 <comm>
  <type>input</type>
  <value>myLocation</value>
 </comm>
 <comm>
  <comm>
   <type>invocation</type>
   <alias>myLocAd</alias>
   <params>
    <param>myLocation</param>
    <param>Tdirectory</param>
    <param>dxs</param>
   </params>
   <service>location</service>
  </comm>
  <comm>
   <type>invocation</type>
   <params>
    <param>target</param>
    <param>
     <comm>
      <type>dxs</type>
      <method>set</method>
```

```
    <value>0</value>
   </comm>
  </param>
 </params>
 <service>cityHall.location</service>
</comm>
<comm>
 <type>invocation</type>
 <params>
  <param>target</param>
  <param>
   <comm>
    <type>dxs</type>
    <method>set</method>
    <value>0</value>
   </comm>
  </param>
 </params>
 <service>teleCo.location</service>
</comm>
<comm>
 <type>invocation</type>
 <params>
  <param>target</param>
  <param>
   <comm>
    <type>dxs</type>
    <method>set</method>
    <value>0</value>
   </comm>
  </param>
 </params>
 <service>coAssociation.location</service>
</comm>
<comm>
 <type>invocation</type>
 <params>
  <param>target</param>
  <param>
   <comm>
    <type>dxs</type>
    <method>set</method>
    <value>0</value>
   </comm>
  </param>
 </params>
 <service>tourismGuide.location</service>
</comm>
<comm>
 <type>invocation</type>
 <alias>pAd</alias>
 <params>
  <param>
   <comm>
    <type>dxs</type>
```

```
      <method>get</method>
      <value>0</value>
     </comm>
    </param>
    <param>Tdirectory</param>
    <param>dxs</param>
   </params>
   <service>pointsOfInterest</service>
  </comm>
 </comm>
 <comm>
  <type>dxs</type>
  <method>wait</method>
  <value>pAd</value>
 </comm>
 <comm>
  <type>output</type>
  <value>
   <comm>
    <type>dxs</type>
    <method>get</method>
    <value>pAd</value>
   </comm>
  </value>
 </comm>
 <comm>
  <type>input</type>
  <value>p2</value>
 </comm>
 <comm>
  <type>dxs</type>
  <method>store</method>
  <alias>p2Ad</alias>
  <value>p2</value>
 </comm>
 <comm>
  <type>invocation</type>
  <params>
   <param>Troute</param>
   <param>myLocAd</param>
   <param>p2Ad</param>
   <param>
    <comm>
     <type>dxs</type>
     <method>get</method>
     <value>0</value>
    </comm>
   </param>
  </params>
  <service>query</service>
 </comm>
</transaction>
```

A.2 Formato em XML Schema

O código seguinte apresenta o formato em XML Schema para a entrada do *Transaction Control Service*.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:simpleType name="T_type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="dxs"/>
   <xs:enumeration value="input"/>
   <xs:enumeration value="invocation"/>
   <xs:enumeration value="output"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="T_service">
  <xs:restriction base="xs:string">
   <xs:enumeration value="cityHall.location"/>
   <xs:enumeration value="coAssociation.location"/>
   <xs:enumeration value="location"/>
   <xs:enumeration value="pointsOfInterest"/>
   <xs:enumeration value="query"/>
   <xs:enumeration value="teleCo.location"/>
   <xs:enumeration value="tourismGuide.location"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="T_method">
  <xs:restriction base="xs:string">
   <xs:enumeration value="get"/>
   <xs:enumeration value="set"/>
   <xs:enumeration value="store"/>
   <xs:enumeration value="wait"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="T_alias">
  <xs:restriction base="xs:string">
   <xs:enumeration value="myLocAd"/>
   <xs:enumeration value="p2Ad"/>
   <xs:enumeration value="pAd"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:complexType name="T_value" mixed="true">
  <xs:sequence>
   <xs:element ref="comm" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="T_transaction">
  <xs:sequence>
   <xs:element ref="comm" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="T_params">
  <xs:sequence>
   <xs:element ref="param" maxOccurs="unbounded"/>
```

```
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="T_param" mixed="true">
  <xs:sequence>
   <xs:element ref="comm" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="T_comm">
  <xs:choice>
   <xs:element ref="comm" maxOccurs="unbounded"/>
   <xs:sequence>
    <xs:element ref="type"/>
    <xs:choice>
     <xs:element ref="value"/>
     <xs:sequence>
      <xs:element ref="alias" minOccurs="0"/>
      <xs:element ref="params"/>
      <xs:element ref="service"/>
     </xs:sequence>
     <xs:sequence>
      <xs:element ref="method"/>
      <xs:element ref="alias" minOccurs="0"/>
      <xs:element ref="value"/>
     </xs:sequence>
    </xs:choice>
   </xs:sequence>
  </xs:choice>
 </xs:complexType>
 <xs:element name="value" type="T_value"/>
 <xs:element name="type" type="T_type"/>
 <xs:element name="transaction" type="T_transaction"/>
 <xs:element name="service" type="T_service"/>
 <xs:element name="params" type="T_params"/>
 <xs:element name="param" type="T_param"/>
 <xs:element name="method" type="T_method"/>
 <xs:element name="comm" type="T_comm"/>
 <xs:element name="alias" type="T_alias"/>
</xs:schema>
```

A.3 Transformador

O código seguinte apresenta o documento XSLT para transformação do documento de entrada do *Transaction Control Service* para código executável. O XSLT apresentado representa o transformador da entrada para a linguagem Java.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:fn="http://www.w3.org/2005/xpath-functions">
 <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
 <xsl:template match="/">
  <body>
```

```
  <p>// Inicializacao<br/>
  loadServices(addrCatalog);<br/>
  DXSService dxs = new DXSService();<br/>
  dxs.start();</p>
  <xsl:for-each select="/transaction/comm">
   <p>
    <xsl:apply-templates select="."/>;
   </p>
  </xsl:for-each>
  <!--   <xsl:apply-templates select="/transaction/comm"/> -->
  <p>// Fechamento</p>
 </body>
</xsl:template>
<xsl:template match="comm">
 <xsl:choose>
  <xsl:when test="type='invocation'">
    <xsl:apply-templates select="alias"/>
    <xsl:value-of select="service"/>(<xsl:apply-templates select="params"/>)
  </xsl:when>
  <xsl:when test="type='input'">
   // Interacao com o usuario - Programador deve definir entrada/saida<br/>
    <xsl:value-of select="value"/> = client.from()
  </xsl:when>
  <xsl:when test="type='output'">
   // Interacao com o usuario - Programador deve definir entrada/saida
   <br/>client.to(<xsl:value-of select="value"/>
    <xsl:apply-templates select="comm"/>)
  </xsl:when>
  <xsl:when test="type='dxs'">
   <xsl:choose>
    <xsl:when test="method='store'">
     <xsl:apply-templates select="alias"/>dxs.store(<xsl:value-of
      select="value"/>)
    </xsl:when>
    <xsl:when test="method='wait'">
      <xsl:apply-templates select="alias"/>dxs.waitItem(<xsl:value-of
       select="value"/>)
    </xsl:when>
    <xsl:when test="method='get'">
      <xsl:apply-templates select="alias"/>dxs.getItem(<xsl:value-of
       select="value"/>)
    </xsl:when>
    <xsl:when test="method='set'">
      <xsl:apply-templates select="alias"/>dxs.setItem(<xsl:value-of
       select="value"/>)
    </xsl:when>
   </xsl:choose>
  </xsl:when>
  <xsl:otherwise>
   // Comandos paralelos<br/>
   <xsl:for-each select="comm">
    <xsl:apply-templates select="."/><xsl:if test="position() != last()">;<br/>
     </xsl:if>
   </xsl:for-each>
  </xsl:otherwise>
```

```
  </xsl:choose>
 </xsl:template>
 <xsl:template match="alias">
  <xsl:value-of select="."/> =
 </xsl:template>
 <xsl:template match="params">
  <xsl:for-each select="param">
   <xsl:choose>
    <xsl:when test="not(comm)">
     <xsl:value-of select="."/>
     <xsl:if test="position() != last()">, </xsl:if>
    </xsl:when>
    <xsl:otherwise>
     <xsl:apply-templates select="comm"/>
     <xsl:if test="position() != last()">, </xsl:if>
    </xsl:otherwise>
   </xsl:choose>
  </xsl:for-each>
 </xsl:template>
</xsl:stylesheet>
```

Apêndice B

Esquemas de Serviços Web Geoespaciais

B.1 WSDL Geocoder não-padrão

O código seguinte apresenta o WSDL para serviços do tipo `Geocoder` não-padrão usados no experimento.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace=
                  "http://localhost:8080/axis/services/Geocoder_BasemapA"
                  xmlns:apachesoap="http://xml.apache.org/xml-soap"
                  xmlns:impl=
                  "http://localhost:8080/axis/services/Geocoder_BasemapA"
                  xmlns:intf=
                  "http://localhost:8080/axis/services/Geocoder_BasemapA"
                  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                  xmlns:tns1="urn:BeanService"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:23:37 EDT)-->
 <wsdl:types>
  <schema targetNamespace="urn:BeanService"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="GeocoderFeatureRequest">
    <sequence>
     <element name="featureType" nillable="true" type="xsd:string"/>
     <element name="query1" nillable="true" type="xsd:string"/>
     <element name="query2" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="GeocoderFeatureResponse">
    <sequence>
```

```
    <element name="builderNumber" nillable="true" type="xsd:string"/>
    <element name="cityName" nillable="true" type="xsd:string"/>
    <element name="countryName" nillable="true" type="xsd:string"/>
    <element name="element" nillable="true" type="xsd:string"/>
    <element name="featureType" nillable="true" type="xsd:string"/>
    <element name="geometry" nillable="true" type="xsd:string"/>
    <element name="landmarkTitle" nillable="true" type="xsd:string"/>
    <element name="lat" type="xsd:double"/>
    <element name="lon" type="xsd:double"/>
    <element name="neighborhood" nillable="true" type="xsd:string"/>
    <element name="organizationName" nillable="true" type="xsd:string"/>
    <element name="postalCode" nillable="true" type="xsd:string"/>
    <element name="stateName" nillable="true" type="xsd:string"/>
    <element name="streetName" nillable="true" type="xsd:string"/>
    <element name="telephoneNumber" nillable="true" type="xsd:string"/>
    <element name="telephoneSubscriber" nillable="true" type="xsd:string"/>
    <element name="x" type="xsd:double"/>
    <element name="y" type="xsd:double"/>
   </sequence>
  </complexType>
  <complexType name="GeocoderFeatureTypeDescriptor">
   <sequence/>
  </complexType>
  <complexType name="GeocoderCapabilities">
   <sequence/>
  </complexType>
 </schema>
</wsdl:types>

  <wsdl:message name="getFeatureResponse">
     <wsdl:part name="getFeatureReturn" type="tns1:GeocoderFeatureResponse"/>
  </wsdl:message>
  <wsdl:message name="describeFeatureTypeResponse">
     <wsdl:part name="describeFeatureTypeReturn"
      type="tns1:GeocoderFeatureTypeDescriptor"/>
  </wsdl:message>
  <wsdl:message name="describeFeatureTypeRequest">
  </wsdl:message>
  <wsdl:message name="getFeatureResponse1">
     <wsdl:part name="getFeatureReturn"
      type="tns1:GeocoderFeatureResponse"/>
  </wsdl:message>
  <wsdl:message name="getCapabilitiesRequest">
  </wsdl:message>
  <wsdl:message name="getFeatureRequest">
     <wsdl:part name="geocoderFeatureRequest"
      type="tns1:GeocoderFeatureRequest"/>
  </wsdl:message>
  <wsdl:message name="getCapabilitiesResponse">
     <wsdl:part name="getCapabilitiesReturn"
      type="tns1:GeocoderCapabilities"/>
  </wsdl:message>
  <wsdl:message name="getFeatureRequest1">
  </wsdl:message>
  <wsdl:portType name="Geocoder_BasemapA">
```

```
    <wsdl:operation name="getFeature"
     parameterOrder="geocoderFeatureRequest">
       <wsdl:input message="impl:getFeatureRequest"
        name="getFeatureRequest"/>
       <wsdl:output message="impl:getFeatureResponse"
        name="getFeatureResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getFeature">
       <wsdl:input message="impl:getFeatureRequest1"
        name="getFeatureRequest1"/>
       <wsdl:output message="impl:getFeatureResponse1"
        name="getFeatureResponse1"/>
    </wsdl:operation>
    <wsdl:operation name="describeFeatureType">
       <wsdl:input message="impl:describeFeatureTypeRequest"
        name="describeFeatureTypeRequest"/>
       <wsdl:output message="impl:describeFeatureTypeResponse"
        name="describeFeatureTypeResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getCapabilities">
       <wsdl:input message="impl:getCapabilitiesRequest"
        name="getCapabilitiesRequest"/>
       <wsdl:output message="impl:getCapabilitiesResponse"
        name="getCapabilitiesResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="Geocoder_BasemapASoapBinding"
   type="impl:Geocoder_BasemapA">
    <wsdlsoap:binding style="rpc"
     transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getFeature">
       <wsdlsoap:operation soapAction=""/>
       <wsdl:input name="getFeatureRequest">
          <wsdlsoap:body
           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
           namespace="http://DefaultNamespace" use="encoded"/>
       </wsdl:input>
       <wsdl:output name="getFeatureResponse">
          <wsdlsoap:body
           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
           namespace="http://localhost:8080/axis/services/Geocoder_BasemapA"
           use="encoded"/>
       </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getFeature">
       <wsdlsoap:operation soapAction=""/>
       <wsdl:input name="getFeatureRequest1">
          <wsdlsoap:body
           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
           namespace="http://DefaultNamespace" use="encoded"/>
       </wsdl:input>
       <wsdl:output name="getFeatureResponse1">
          <wsdlsoap:body
           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
           namespace="http://localhost:8080/axis/services/Geocoder_BasemapA"
```

```
            use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="describeFeatureType">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="describeFeatureTypeRequest">
            <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://DefaultNamespace" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="describeFeatureTypeResponse">
            <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://localhost:8080/axis/services/Geocoder_BasemapA"
             use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getCapabilities">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getCapabilitiesRequest">
            <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://DefaultNamespace" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getCapabilitiesResponse">
            <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://localhost:8080/axis/services/Geocoder_BasemapA"
             use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="Geocoder_BasemapAService">
      <wsdl:port
       binding="impl:Geocoder_BasemapASoapBinding"
       name="Geocoder_BasemapA">
         <wsdlsoap:address
          location="http://localhost:8080/axis/services/Geocoder_BasemapA"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

B.2 WSDL Directory não-padrão

O código seguinte apresenta o WSDL para serviços do tipo `Directory` não-padrão usados no experimento.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/Directory.jws"
 xmlns:apachesoap="http://xml.apache.org/xml-soap"
 xmlns:impl="http://localhost:8080/axis/Directory.jws"
 xmlns:intf="http://localhost:8080/axis/Directory.jws"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
```

```
 xmlns:tns1="http://DefaultNamespace" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:23:37 EDT)-->
 <wsdl:types>
  <schema targetNamespace="http://DefaultNamespace"
   xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="DirectoryFeatureRequest">
    <sequence>
     <element name="featureType" nillable="true" type="xsd:string"/>
     <element name="query1" nillable="true" type="xsd:string"/>
     <element name="query2" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="DirectoryFeatureResponse">
    <sequence>
     <element name="builderNumber" nillable="true" type="xsd:string"/>
     <element name="cityName" nillable="true" type="xsd:string"/>
     <element name="countryName" nillable="true" type="xsd:string"/>
     <element name="element" nillable="true" type="xsd:string"/>
     <element name="featureType" nillable="true" type="xsd:string"/>
     <element name="geometry" nillable="true" type="xsd:string"/>
     <element name="landmarkTitle" nillable="true" type="xsd:string"/>
     <element name="lat" type="xsd:double"/>
     <element name="lon" type="xsd:double"/>
     <element name="neighborhood" nillable="true" type="xsd:string"/>
     <element name="organizationName" nillable="true" type="xsd:string"/>
     <element name="postalCode" nillable="true" type="xsd:string"/>
     <element name="stateName" nillable="true" type="xsd:string"/>
     <element name="streetName" nillable="true" type="xsd:string"/>
     <element name="telephoneNumber" nillable="true" type="xsd:string"/>
     <element name="telephoneSubscriber" nillable="true" type="xsd:string"/>
     <element name="x" type="xsd:double"/>
     <element name="y" type="xsd:double"/>
    </sequence>
   </complexType>
   <complexType name="DirectoryFeatureTypeDescriptor">
    <sequence/>
   </complexType>
   <complexType name="DirectoryCapabilities">
    <sequence/>
   </complexType>
  </schema>
 </wsdl:types>
   <wsdl:message name="getCapabilitiesRequest">
   </wsdl:message>
   <wsdl:message name="describeFeatureTypeRequest">
   </wsdl:message>
   <wsdl:message name="getFeatureResponse1">
      <wsdl:part name="getFeatureReturn"
       type="tns1:DirectoryFeatureResponse"/>
   </wsdl:message>
   <wsdl:message name="getFeatureRequest1">
```

```
</wsdl:message>
<wsdl:message name="describeFeatureTypeResponse">
   <wsdl:part name="describeFeatureTypeReturn"
    type="tns1:DirectoryFeatureTypeDescriptor"/>
</wsdl:message>
<wsdl:message name="getFeatureResponse">
   <wsdl:part name="getFeatureReturn"
    type="tns1:DirectoryFeatureResponse"/>
</wsdl:message>
<wsdl:message name="getFeatureRequest">
   <wsdl:part name="geocoderFeatureRequest"
    type="tns1:DirectoryFeatureRequest"/>
</wsdl:message>
<wsdl:message name="getCapabilitiesResponse">
   <wsdl:part name="getCapabilitiesReturn"
    type="tns1:DirectoryCapabilities"/>
</wsdl:message>
<wsdl:portType name="Directory">
   <wsdl:operation name="getFeature"
    parameterOrder="geocoderFeatureRequest">
      <wsdl:input message="impl:getFeatureRequest"
       name="getFeatureRequest"/>
      <wsdl:output message="impl:getFeatureResponse"
       name="getFeatureResponse"/>
   </wsdl:operation>
   <wsdl:operation name="getFeature">
      <wsdl:input message="impl:getFeatureRequest1"
       name="getFeatureRequest1"/>
      <wsdl:output message="impl:getFeatureResponse1"
       name="getFeatureResponse1"/>
   </wsdl:operation>
   <wsdl:operation name="describeFeatureType">
      <wsdl:input message="impl:describeFeatureTypeRequest"
       name="describeFeatureTypeRequest"/>
      <wsdl:output message="impl:describeFeatureTypeResponse"
       name="describeFeatureTypeResponse"/>
   </wsdl:operation>
   <wsdl:operation name="getCapabilities">
      <wsdl:input message="impl:getCapabilitiesRequest"
       name="getCapabilitiesRequest"/>
      <wsdl:output message="impl:getCapabilitiesResponse"
       name="getCapabilitiesResponse"/>
   </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="DirectorySoapBinding" type="impl:Directory">
   <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="getFeature">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getFeatureRequest">
         <wsdlsoap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://DefaultNamespace" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="getFeatureResponse">
```

```
            <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://localhost:8080/axis/Directory.jws"
             use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getFeature">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getFeatureRequest1">
            <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://DefaultNamespace" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getFeatureResponse1">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://localhost:8080/axis/Directory.jws" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="describeFeatureType">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="describeFeatureTypeRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://DefaultNamespace" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="describeFeatureTypeResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://localhost:8080/axis/Directory.jws" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getCapabilities">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getCapabilitiesRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://DefaultNamespace" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getCapabilitiesResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://localhost:8080/axis/Directory.jws" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="DirectoryService">
      <wsdl:port binding="impl:DirectorySoapBinding" name="Directory">
         <wsdlsoap:address location="http://localhost:8080/axis/Directory.jws"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

B.3 WSDL WFS

O código seguinte apresenta o WSDL para serviços do tipo `Web Feature Service` usados no experimento.

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

```
<wsdl:definitions targetNamespace="http://localhost:8080/axis/wfs.jws"
 xmlns:apachesoap="http://xml.apache.org/xml-soap"
 xmlns:impl="http://localhost:8080/axis/wfs.jws"
 xmlns:intf="http://localhost:8080/axis/wfs.jws"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:tns1="http://DefaultNamespace"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (06:10:28 EDT)-->
   <wsdl:documentation xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:description>
         HTTP/1.1 protocol bindings for WFS interfaces.
      </dc:description>
      <dc:date>2004-06-07</dc:date>
   </wsdl:documentation>

   <wsdl:import namespace="http://www.opengis.net/wfs/requests"
                location="./wfs-xml-interfaces.wsdl"/>

   <wsdl:binding name="wfs-POST" type="wfs-req:wfs">
      <wsdl:documentation>
         wfs interface bound to the HTTP/1.1 POST method.
      </wsdl:documentation>
      <http:binding verb="POST"/>
      <wsdl:operation name="wfs.getCapabilities">
         <http:operation location="wfs/http"/>
         <wsdl:input>
            <mime:mimeXml/>
         </wsdl:input>
         <wsdl:output>
            <mime:mimeXml/>
         </wsdl:output>
         <wsdl:fault name="ServiceExceptionReport">
            <mime:mimeXml/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="wfs.describeFeatureType">
         <http:operation location="wfs/http"/>
         <wsdl:input>
            <mime:mimeXml/>
         </wsdl:input>
         <wsdl:output>
            <mime:mimeXml/>
         </wsdl:output>
         <wsdl:fault name="ServiceExceptionReport">
            <mime:mimeXml/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="wfs.getFeature">
         <http:operation location="wfs/http"/>
         <wsdl:input>
            <mime:mimeXml/>
         </wsdl:input>
```

```
      <wsdl:output>
         <mime:mimeXml/>
      </wsdl:output>
      <wsdl:fault name="ServiceExceptionReport">
         <mime:mimeXml/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="wfs.getFeatureWithLock">
      <http:operation location="wfs/http"/>
      <wsdl:input>
         <mime:mimeXml/>
      </wsdl:input>
      <wsdl:output>
         <mime:mimeXml/>
      </wsdl:output>
      <wsdl:fault name="ServiceExceptionReport">
         <mime:mimeXml/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="wfs.getGMLObject">
      <http:operation location="wfs/http"/>
      <wsdl:input>
         <mime:mimeXml/>
      </wsdl:input>
      <wsdl:output>
         <mime:mimeXml/>
      </wsdl:output>
      <wsdl:fault name="ServiceExceptionReport">
         <mime:mimeXml/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="wfs.lockFeature">
      <http:operation location="wfs/http"/>
      <wsdl:input>
         <mime:mimeXml/>
      </wsdl:input>
      <wsdl:output>
         <mime:mimeXml/>
      </wsdl:output>
      <wsdl:fault name="ServiceExceptionReport">
         <mime:mimeXml/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="wfs.transaction">
      <http:operation location="wfs/http"/>
      <wsdl:input>
         <mime:mimeXml/>
      </wsdl:input>
      <wsdl:output>
         <mime:mimeXml/>
      </wsdl:output>
      <wsdl:fault name="ServiceExceptionReport">
         <mime:mimeXml/>
      </wsdl:fault>
   </wsdl:operation>
```

```
   </wsdl:binding>
</wsdl:definitions>
```

B.4 WSDL WMS

O código seguinte apresenta o WSDL para serviços do tipo `Web Map Service` usados no experimento.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/wms.jws"
 xmlns:apachesoap="http://xml.apache.org/xml-soap"
 xmlns:impl="http://localhost:8080/axis/wms.jws"
 xmlns:intf="http://localhost:8080/axis/wms.jws"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:tns1="http://DefaultNamespace"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:35:12 EDT)-->
<wsdl:documentation>
</wsdl:documentation>
<wsdl:import namespace="http://www.opengis.net/wms/requests"
 location="./wms-xml-interfaces.wsdl"/>
<wsdl:binding name="WMS_SOAP_Binding"
type="wms-req:WMS_XML_Port">
<soap:binding style="document"
 transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="GetCapabilities">
<soap:operation/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<wsdl:fault name="exception">
<soap:fault name="exception" use="literal"/>
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="GetMap">
<soap:operation/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<mime:content type="image/*"/>
</wsdl:output>
<wsdl:fault name="exception">
<soap:fault name="exception" use="literal"/>
</wsdl:fault>
</wsdl:operation>
</wsdl:binding>
</wsdl:definitions>
```

B.5 Schema Route não-padrão

O código seguinte apresenta o formato em XML Schema para os tipos do `Route` não-padrão usados na avaliação. O formato é baseado no serviço `Route` padrão e altera principalmente os valores *default*.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/xls"
 xmlns:gml="http://www.opengis.net/gml"
 xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:xls="http://www.opengis.net/xls"
 elementFormDefault="qualified" version="1.1">
<import namespace="http://www.opengis.net/gml" schemaLocation="gml4xls.xsd"/>
<include schemaLocation="XLS.xsd"/>

<element name="DetermineRouteRequest"
 type="xls:DetermineRouteRequestType"
 substitutionGroup="xls:_RequestParameters">
<annotation>
<documentation>Specifies the Determine Route request
 parameters.</documentation>
</annotation>
</element>
<complexType name="DetermineRouteRequestType">
<annotation>
<documentation>Defines the Determine Route request
 parameters.</documentation>
</annotation>
<complexContent>
<extension base="xls:AbstractRequestParametersType">
<sequence>
<choice>
<element ref="xls:RouteHandle">
<annotation>
<documentation>Reference to a proviously determined
 route stored at the Route Determination Service
 server.</documentation>
</annotation>
</element>
<element ref="xls:RoutePlan"/>
</choice>
<element ref="xls:RouteInstructionsRequest" minOccurs="0">
<annotation>
<documentation>Request parameters for turn-by-turn
 route directions and advisories formatted for
 presentation.</documentation>
</annotation>
</element>
<element ref="xls:RouteGeometryRequest" minOccurs="0">
<annotation>
<documentation>Request parameters for route
 geometry.</documentation>
</annotation>
</element>
```

```
<element ref="xls:RouteMapRequest" minOccurs="0"/>
</sequence>
<attribute name="provideRouteHandle" type="boolean"
 use="optional" default="false">
<annotation>
<documentation>Requests the return of a route
 handle.</documentation>
</annotation>
</attribute>
<attribute name="distanceUnit" type="xls:DistanceUnitType"
 use="optional" default="M">
<annotation>
<documentation>Specifies the unit for measuring
 distance.</documentation>
</annotation>
</attribute>
</extension>
</complexContent>
</complexType>
<element name="RoutePlan" type="xls:RoutePlanType">
<annotation>
<documentation>The criteria upon which a route is
 determined.</documentation>
</annotation>
</element>
<complexType name="RoutePlanType">
<annotation>
<documentation>Defines the criteria upon which a route
 is determined.</documentation>
</annotation>
<sequence>
<element ref="xls:RoutePreference"/>
<element ref="xls:WayPointList"/>
<element ref="xls:AvoidList" minOccurs="0"/>
</sequence>
<attribute name="useRealTimeTraffic" type="boolean"
 use="optional" default="true">
<annotation>
<documentation>Specifies whether to use real time traffic
 information when determining the best route.</documentation>
</annotation>
</attribute>
<attribute name="expectedStartTime" type="dateTime" use="optional">
<annotation>
<documentation>Specifies the date and time at which travel
 is expected to begin.  Specified in the format YYYY-MM-DD
 HH:MM.  Defaults to current date and time.</documentation>
</annotation>
</attribute>
<attribute name="expectedEndTime" type="dateTime" use="optional">
<annotation>
<documentation>Specifies the date and time at which travel
 is expected to end.  The format for the end time is
 specified as Duration</documentation>
</annotation>
```

```
</attribute>
</complexType>
<element name="AvoidList" type="xls:AvoidListType">
<annotation>
<documentation>The list of areas, locations, and types of
 features in which the route should avoid passing
 through.</documentation>
</annotation>
</element>
<complexType name="AvoidListType">
<annotation>
<documentation>Defines the list of areas, locations, and
 types of features in which the route should avoid
 passing through.</documentation>
</annotation>
<sequence>
<element ref="xls:AOI" minOccurs="0" maxOccurs="unbounded">
<annotation>
<documentation>List of geographic areas to
 avoid.</documentation>
</annotation>
</element>
<element ref="xls:_Location" minOccurs="0" maxOccurs="unbounded">
<annotation>
<documentation>List of locations to
 avoid.</documentation>
</annotation>
</element>
<element ref="xls:AvoidFeature" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexType>
<element name="RouteMapRequest" type="xls:RouteMapRequestType">
<annotation>
<documentation>The request parameters for route maps.</documentation>
</annotation>
</element>
<complexType name="RouteMapRequestType">
<annotation>
<documentation>Defines the request parameters for route maps.</documentation>
</annotation>
<sequence>
<element name="Output" type="xls:RouteMapOutputType" maxOccurs="unbounded"/>
</sequence>
</complexType>
<complexType name="RouteMapOutputType">
<annotation>
<documentation>Defines the rendered route map output parameters.</documentation>
</annotation>
<sequence>
<element name="BBoxContext" type="gml:EnvelopeType" minOccurs="0">
<annotation>
<documentation>Rectangular area to be displayed in the
 rendered map. If not specified, defaults to full
 route.</documentation>
</annotation>
```

```
<!-- type="xls:BoxType" -->
</element>
</sequence>
<attribute name="width" type="nonNegativeInteger">
<annotation>
<documentation>pixel width of the resulting map</documentation>
</annotation>
</attribute>
<attribute name="height" type="nonNegativeInteger">
<annotation>
<documentation>pixel height of the resulting map</documentation>
</annotation>
</attribute>
<attribute name="format" type="string">
<annotation>
<documentation>mime type describing the encoding</documentation>
</annotation>
</attribute>
<attribute name="BGcolor" type="string" use="optional"/>
<attribute name="transparent" type="boolean" use="optional"/>
<attribute name="style" type="xls:RouteMapStyleType" use="optional"/>
</complexType>
<simpleType name="RouteMapStyleType">
<annotation>
<documentation>A route map can be either an overview
 or a maneuver</documentation>
</annotation>
<restriction base="string">
<enumeration value="Overview">
<annotation>
<documentation>Used to describe the map showing
 the full route</documentation>
</annotation>
</enumeration>
<enumeration value="Maneuver">
<annotation>
<documentation>Used to describe the map showing a
 particular maneuver (often the maneuver corresponds
 to a single instruction)</documentation>
</annotation>
</enumeration>
</restriction>
</simpleType>
<element name="RouteInstructionsRequest"
 type="xls:RouteInstructionsRequestType">
<annotation>
<documentation>The request parameters for turn-by-turn
 route instructions and travel advisories formatted for
 presentation.</documentation>
</annotation>
</element>
<complexType name="RouteInstructionsRequestType">
<annotation>
<documentation>Defines the request parameters for
 turn-by-turn route instructions and travel
```

```
 advisories formatted for presentation.</documentation>
</annotation>
<attribute name="format" type="string"
 use="optional" default="text/plain">
<annotation>
<documentation>The preferred format of the route
 instructions, specified as a mime type.
 </documentation>
</annotation>
</attribute>
<attribute name="provideGeometry" type="boolean"
 use="optional" default="false"/>
<attribute name="provideBoundingBox" type="boolean"
 use="optional" default="false"/>
</complexType>
<element name="RouteGeometryRequest"
 type="xls:RouteGeometryRequestType">
<annotation>
<documentation>The request parameters for route
 geometry.</documentation>
</annotation>
</element>
<complexType name="RouteGeometryRequestType">
<annotation>
<documentation>Defines the request parameters for
 route geometry.</documentation>
</annotation>
<sequence>
<element name="BoundingBox"
 type="gml:EnvelopeType" minOccurs="0">
<annotation>
<documentation>Rectangular area of route for which
 the geometry is requested.  If not specified,
 defaults to full route.</documentation>
</annotation>
<!-- type="xls:BoxType" -->
</element>
</sequence>
<attribute name="scale" type="positiveInteger"
 use="optional" default="1">
<annotation>
<documentation>Maximum scale at which the route will
 be displayed. Expressed as a ratio of world units to
 a device unit.  For example 1:50000 would be specified
 as 50000.</documentation>
</annotation>
</attribute>
<attribute name="provideStartingPortion"
 type="boolean" use="optional" default="false">
<annotation>
<documentation>If true, return the geometry of the
 starting portion of the route contained within the
 specified bounding area, up to the specified maximum
 number of points.  If false, return the geometry of
 the complete route contained within the specified
```

```
 area, reducing the accuracy of the geometry as
 necessary to not exceed the specified maximum
 number of points.</documentation>
</annotation>
</attribute>
<attribute name="maxPoints"
 type="positiveInteger" use="optional" default="100">
<annotation>
<documentation>Maximum number of geometric points
to return.</documentation>
</annotation>
</attribute>
</complexType>

<element name="DetermineRouteResponse"
 type="xls:DetermineRouteResponseType"
 substitutionGroup="xls:_ResponseParameters">
<annotation>
<documentation>Specifies the Determine Route response
 parameters.</documentation>
</annotation>
</element>
<complexType name="DetermineRouteResponseType">
<annotation>
<documentation>Defines the Determine Route response
 parameters.</documentation>
</annotation>
<complexContent>
<extension base="xls:AbstractResponseParametersType">
<sequence>
<element ref="xls:RouteHandle" minOccurs="0">
<annotation>
<documentation>Reference to the route stored at
 the Route Determination Service server.</documentation>
</annotation>
</element>
<element ref="xls:RouteSummary">
<annotation>
<documentation>Response for requested
 route summary.</documentation>
</annotation>
</element>
<element ref="xls:RouteGeometry" minOccurs="0">
<annotation>
<documentation>Response for requested
 route geometry.</documentation>
</annotation>
</element>
<element ref="xls:RouteInstructionsList" minOccurs="0">
<annotation>
<documentation>Response for requested
 route instructions.</documentation>
</annotation>
</element>
<element ref="xls:RouteMap" minOccurs="0" maxOccurs="unbounded">
```

```
<annotation>
<documentation>Response list for
 requested route maps.</documentation>
</annotation>
</element>
</sequence>
</extension>
</complexContent>
</complexType>
<element name="RouteMap" type="xls:RouteMapType">
<annotation>
<documentation>A route map.</documentation>
</annotation>
</element>
<complexType name="RouteMapType">
<annotation>
<documentation>Defines a route map.</documentation>
</annotation>
<complexContent>
<extension base="xls:MapType">
<attribute name="description" type="string">
<annotation>
<documentation>Allows the route instruction to be
 matched with a RouteMapType. For
 example "maneuver 1"</documentation>
</annotation>
</attribute>
</extension>
</complexContent>
</complexType>
<element name="AvoidFeature" type="xls:AvoidFeatureType">
<annotation>
<documentation>Type of feature to avoid when
 determining the route.</documentation>
</annotation>
</element>
<simpleType name="AvoidFeatureType">
<annotation>
<documentation>Enumeration of types of features
 to avoid when determining the route.</documentation>
</annotation>
<restriction base="string">
<enumeration value="Highway">
<annotation>
<documentation>Minimize the use of highways.</documentation>
</annotation>
</enumeration>
<enumeration value="Tollway">
<annotation>
<documentation>Minimize tolls.</documentation>
</annotation>
</enumeration>
</restriction>
</simpleType>
<simpleType name="RoutePreferenceType">
```

```
<annotation>
<documentation>Enumeration of preferences
 to be taken into consideration when determining
 the route.</documentation>
</annotation>
<restriction base="string">
<enumeration value="Fastest">
<annotation>
<documentation>Minimize the travel
 time by vehicle.</documentation>
</annotation>
</enumeration>
<enumeration value="Shortest">
<annotation>
<documentation>Minimize the travel
 distance by vehicle.</documentation>
</annotation>
</enumeration>
<enumeration value="Pedestrian">
<annotation>
<documentation>Best route by foot.</documentation>
</annotation>
</enumeration>
</restriction>
</simpleType>
<element name="RoutePreference" type="xls:RoutePreferenceType">
<annotation>
<documentation>Preference to be taken into
 consideration when determining the route.</documentation>
</annotation>
</element>
</schema>
```