Interoperabilidade, Processamento e Análise Visual de Dados Geográficos

Bruno Manuel Gameiro Simões

Orientador: Prof. Dr. Luís Miguel Rato

Tese apresentada à Universidade de Évora, como parte dos requisitos para obtenção do título de Mestrado em Engenharia Informática

Évora, 23 de Julho de 2009

Interoperabilidade, processamento e análise visual de dados geográficos

Bruno Manuel Gameiro Simões

Orientador: Prof. Dr. Luís Miguel Rato

Agradecimentos

Embora uma dissertação seja, pela sua finalidade académica, um trabalho individual, há contributos de natureza diversa que não podem nem devem deixar de ser realçados. Por essa razão, desejo expressar os meus sinceros agradecimentos:

Ao Prof. Dr. Luís Rato, pela disponibilidade, assim como pelas críticas e sugestões feitas durante a orientação.

À Fondazione Graphitech, pelo reconhecimento e suporte, sem o qual esta dissertação dificilmente teria seguido o rumo pretendido.

Ao Dr. Giuseppe Conti, cuja familiaridade com as necessidades e ideias do tema foram úteis durante a fase inicial de planeamento desta obra. Pelo profissionalismo exemplar, pelo apoio, pelas críticas e sugestões, e pela sua amizade.

Ao William Wright, co-autor de "GeoTime Information Visualization" pela sua disponibilidade, contributo e interesse manifestado.

Aos meus pais, pelo estímulo e apoio incondicional desde a primeira hora; pela paciência e grande amizade com que sempre me ouviram, pela e sensatez com que sempre me ajudaram.

À Cindy Silva, pelas inúmeras trocas de impressões e comentários ao trabalho. Acima de tudo, pela compreensão, suporte e ternura sempre manifestadas apesar do 'débito' de atenção revelado ao longo destes meses.

Ao júri, pelas críticas e sugestões feitas ao conteúdo e a apresentação desta dissertação.

Ao Giacomo Zandonini, pelas questões filosóficas levantadas ao longo destes meses, que me obrigaram a reflectir mais aprofundadamente sobre determinados aspectos e pela sua amizade.

Por último gostaria de estender os meus agradecimentos a todos aqueles de uma forma ou de outra (fornecendo ideias e/ou criticando), foram ajudando anonimamente nos inúmeros problemas.

Tabela de Conteúdo

Lista de Figuras

Lista de Tabelas

Glossário

Application Programming Interface (API)	Interface com funcionalidades específicas para o desenvolvimento de determinado tipo de aplicações, permitindo normalmente o acesso a níveis mais baixos do sistema através do uso de rotinas pré-programadas.
Atributo	Propriedade de uma entidade, normalmente usada para referir uma qualificação não espacial de uma entidade georreferenciada. Por exemplo, um código descritivo indicando o que a entidade representa ou como deve ser visualizada.
Booleano	Relativo à álgebra booleana, em que todos os valores são representados apenas por dois valores: zero ou um, sim ou não, verdadeiro ou falso, ligado ou desligado, etc.
Camada	Forma de organização de dados georreferenciados por tipo num mapa. Por exemplo, uma camada poderá conter a densidade populacional, outra camada as cidades, e outras os países e continentes, simplificando assim a criação e actualização de mapas. As camadas podem ser vistas como abstracções que permitem ao utilizador a visualização e análise da informação por categoria ou em conjunto (Panagopoulos, 1999).
Característica geográfica	Entidade geográfica, tal como um edifício ou um rio, extraída de um mapa ou obtida através de levantamento topográfico.
Cartografia	Ciência que, segundo determinados sistemas de projecção e uma determinada escala, representa a totalidade ou parte da superfície terrestre num plano, organizando e permitindo a comunicação de informação geográfica em forma gráfica ou digital. Pode incluir todos os estados desde a aquisição de dados até à sua apresentação e uso (Panagopoulos, 1999).
CDG	Conjunto de dados geográficos. Equivalente ao "dataset" em inglês.
Chave composta	Chave formada por várias colunas de uma tabela.
Chave estrangeira (FK)	Coluna numa tabela do banco de dados relacional cujos valores são extraídos de valores de uma chave primária localizada numa outra tabela.
Chave primária (PK)	Coluna de uma tabela que é exclusivamente diferente em todas as linhas dessa mesma tabela.
Cliente	Componente de software capaz de invocar e apresentar os dados de uma *operação* num servidor.

Coordenadas Cartesianas	Sistema de referência posicional no qual a localização é medida em dois ou três eixos ortogonais (perpendiculares). Toda a localização pode ser definida de modo único pelas suas coordenadas X, Y e Z. Qualquer unidade de medida pode ser utilizada, e.g. metros, pés, milhas. As coordenadas cartesianas diferem das coordenadas latitude e longitude por estas últimas compreenderem um sistema de referência esférico em vez de planar (Panagopoulos, 1999).
Dados	Representação de factos, conceitos ou instruções de modo formal e apropriado à sua comunicação, interpretação ou processamento (Panagopoulos, 1999).
Dados Geográficos	Estes dados são um caso particular de dados espaciais pois apenas dizem respeito à Terra, e são geralmente caracterizados por terem duas componentes fundamentais: o registo de determinado fenómeno, como por exemplo, uma dimensão física (a população de uma cidade, a largura de uma dada estrada) ou uma classe (tipo de rocha, tipo de vegetação, nome de uma cidade), e a localização espacial do fenómeno. A localização é geralmente especificada com a referência a um sistema de coordenadas comum como a latitude e a longitude (Panagopoulos, 1999). Eventualmente os dados podem estar associados a uma característica geográfica invés de uma posição.
Dados Georreferenciados	Dados que dizem respeito a determinadas localizações na superfície terrestre. Os dados georreferenciados podem ser vistos como uma extensão aos dados geográficos, pois não tem necessariamente que descrever características ou acontecimentos relativos à Terra (Por exemplo uma lenda que ocorre em determinada cidade).
Digitalização	Processo de conversão de dados, existentes sob a forma de mapas em papel, desenhos ou fotografias aéreas, para formato digital via (1) decalque manual de mapas com um cursor transparente com guias (puck) auxiliado por entradas de coordenadas ou (2) aquisição digital através de um scanner e subsequente conversão automática realizada por software que traduz a informação raster para informação vectorial (Panagopoulos, 1999).
Digital Elevation Model (DEM)	Ver MDE.
Digital Terrain Model (DTM)	Ver MDT.
Escala	Proporção ou razão entre um valor de medida contido num mapa e o valor real correspondente. A escala de um mapa é geralmente expressa sob a forma de razão, isto é, 1:50 000, significando que a cada unidade de medida do mapa correspondem 50 000 unidades de medida reais (Panagopoulos, 1999).
Exactidão	Proximidade dos resultados de observações, cálculos ou estimativas em relação ao valor verdadeiro ou ao valor aceite como sendo verdadeiro. Exactidão não tem o mesmo significado que precisão. A precisão tem como base o desvio-padrão de uma série de repetições da mesma análise.

Exactidão Absoluta	Correspondência exacta entre a localização das características geográficas num mapa e a sua posição na terra.
Exactidão do Atributo	Diferença entre a informação registada em tabelas de uma base de dados ou num mapa e as características geográficas reais representadas. Por exemplo, no caso de mapas que incluem nomes de ruas como atributos, a percentagem de nomes correctos constituiria o grau de exactidão (Panagopoulos, 1999).
Exactidão Relativa	Diferença entre a forma de características geográficas representadas num mapa e dessas mesmas características *in loco* se posicionam relativamente uma às outras. Um sistema de medida pode empregar um erro sistemático, com a consequente imprecisão de resultados, preservando, no entanto, as relações (Panagopoulos, 1999).
Extensible Markup Language (XML)	Uma forma flexível para a representação de informação, assim como para a sua partilha na World Wide Web, nas intranets, e em qualquer outro lugar. O XML é uma recomendação formal da World Wide Web Consortium (W3C), e uma linguagem similar à das páginas da Web, o Hypertext Markup Language (Uchoa & Ferreira, 2004).
Geocodificação	Geocodificar é a transformação entre o espaço do modelo e o espaço da Terra. A geocodificação vem ao encontro de necessidades comuns em Sistemas de Informação Geográfica (SIG) tais como encontrar as coordenadas geográficas num mapa dado um endereço físico (postal).
Geografia	O termo Geografia, que no latim é geographia, vem do grego $\gamma\varepsilon\omega\gamma\rho\alpha\varphi\iota\alpha$ que é o somatório de $\gamma\eta$ - terra e grafia – $\gamma\rho\alpha\varphi\iota\alpha$ e que traduzido literalmente significa "descrever ou escrever sobre a Terra". Geografia é o estudo da Terra e das suas terras, características, habitantes e fenómenos. Geógrafos estudam a distribuição espacial e temporal de fenómenos, processos e característica, assim como a interacção de humanos e do ambiente.
Geomática	A Geomática é uma ciência e tecnologia interdisciplinar, que envolve geografia, topografia, cartografia, hidrografia, geodesia, matemática, computação entre outras disciplinas, num processo de obtenção, análise, interpretação, distribuição e uso da informação espacial.
Geoprocessamento	O sufixo "processamento", de geoprocessamento, vem do latim processus, que significa "andar avante", "progresso". Os vocábulos latinos processus e progressus têm o mesmo significado, que é "andar avante", "avançar". Assim, pode-se acreditar que o termo geoprocessamento, que surgiu do sentido de processamento de dados georreferenciados, significa introduzir um processo que traga um progresso, um andar avante, na grafia ou representação da Terra. Não é somente representar, mas é associar a esse acto um novo olhar sobre o espaço, a um ganho de informação.
Georreferenciar	Georreferenciar é tornar as coordenadas conhecidas num dado sistema de referência.
GPS	Global Positioning System. Ver Sistema Global de Posicionamento.

Hiperligação	Uma *hiperligação* ou *link* é uma referência num documento em hipertexto a outro documento ou recurso. Como tal, pode-se vê-la como análoga a uma citação na literatura. Ao contrário desta, no entanto, a hiperligação pode ser combinada com uma rede de dados e um protocolo de acesso adequado e assim ser usada para ter acesso directo ao recurso referenciado (Wikipédia, 2008).
Hipertexto	Hipertexto é o termo que remete a um texto em formato digital, ao qual se agrega outros conjuntos de informação na forma de blocos de textos, imagens ou sons, cujo acesso se dá através de referências específicas denominadas *hiperligações* (Ver Hiperligação).
HTTP	Hypertext Transfer Protocol. Ver Protocolo de Transferência de Hipertexto
Identificador Geográfico	Referência geoespacial sob a forma de um topónimo ou código que identifica uma localização.
Informação Espacial	Informação que tem uma referência a uma localização espacial, seja esta num ponto qualquer do universo ou num caso mais concreto como a superfície da Terra.
Informação Geográfica	Informação contida num mapa, digital ou não, que regista a posição física (definida em termos de ponto, área ou volume) e a forma de um elemento geográfico na Terra. Trata-se de um caso particular da Informação Espacial (Panagopoulos, 1999).
Interface	Conjunto de nomes de operações que caracterizam o comportamento de uma entidade.
Internet	Rede constituída por milhares de outras redes, interligadas entre si criando uma única rede virtual à escala mundial. Tem evoluído ao longo dos anos de uma forma anárquica e, como qualquer outra rede de comunicação, é constituída por um conjunto de equipamentos terminais com computadores, alguns tipos de telefones, PDAs, etc., ligados por um meio de transmissão.
Latitude	Primeira componente de um sistema de coordenadas usado para registar posições na superfície da Terra. A latitude é a distância angular entre um ponto qualquer da superfície terrestre e a linha do Equador.
Longitude	Segunda componente de um sistema de coordenadas usado para registar posições na superfície da Terra. A longitude de um lugar é a distância angular entre um ponto qualquer da superfície terrestre e o meridiano inicial ou de origem.
Mapa	Representação gráfica de toda ou parte da superfície da Terra ou do Universo, e de fenómenos, concretos ou abstractos, aí localizados (Panagopoulos, 1999).
Mapa Cartográfico	Colecção de informação, em forma digital ou em papel, que descreve a distribuição espacial das características geográficas à custa de simbologia normalizada (Panagopoulos, 1999).

Metadados	Dados acerca de dados. Por exemplo, os metadados de um mapa digital são as informações acerca da escala, data de revisão, autor, exactidão e outras informações pertinentes (Panagopoulos, 1999).
Modelo Digital de Elevação (MDE)	Qualquer conjunto de dados em suporte numérico, que para uma dada zona, permita associar a qualquer ponto definido sobre o plano cartográfico um valor correspondente à sua altitude. Por outras palavras, é uma representação matemática contínua da distribuição espacial das variações de altitude numa área. É um modelo construído a partir das curvas de nível e/ou pontos altimétricos.
Modelo Digital de Terreno (MDT)	Método de transformação de dados sobre a variação contínua do relevo da superfície terrestre, usando curvas de nível ou representações gráficas tridimensionais (MDE - Modelo Digital de Elevação), e à qual se sobrepõem outros níveis de informação, como por exemplo, informação acerca do uso dos solos, vegetação, estradas, recursos hídricos, etc (Panagopoulos, 1999).
Modelo OSI	O Modelo de Referência OSI ou Modelo OSI é uma descrição abstracta para definir formalmente uma forma comum de conectar computadores. Foi desenvolvido como parte da iniciativa Open Systems Interconnection (OSI) ou Interconexão de Sistemas Abertos. Na sua forma mais básica, divide a arquitectura de rede em sete camadas que, começando do topo, são as camadas de Aplicação, Apresentação, Sessão, Transporte, Rede, de Enlace ou Ligação de Dados, e Físicas.
Mosaico	Subconjunto de uma base de dados SIG que contém informação sobre uma área restrita do mapa digital global. Estes mosaicos (subconjuntos) constituem uma forma eficaz de dividir um mapa contínuo em unidades que podem ser facilmente criadas, editadas e analisadas (Panagopoulos, 1999).
Namespace	Namespaces é uma maneira de superar conflitos entre nomes de elementos num documento ou programa, isto é, um namespace é um espaço ou uma região, dentro de um programa ou documento, onde um nome (variável, função, etc.) é considerado válido. Namespace surge numa tentativa de estruturar nomes de elementos definidos pelo utilizador de modo a evitar que possa acontecer que num mesmo documento exista a coincidência de se utilizar o mesmo elemento para duas situações distintas com semânticas diferentes.
Open DataBase Connectivity (ODBC)	É uma interface para aceder a uma base de dados. Com o uso de relatórios ODBC num programa, pode-se aceder a arquivos em diversas bases de dados, inclusive Access, dBase, DB2, Excel, e Text. Além do software ODBC, é necessário um módulo ou driver separado para cada base de dados a ser acedida.
Padrão Internacional	Modelo de referência com aceitação e uso internacional numa determinada área ou domínio.
Plataforma	É uma expressão utilizada para denominar a tecnologia empregada em determinada infra-estrutura de Tecnologia da Informação ou telecomunicações, garantindo facilidade de integração dos diversos elementos dessa infra-estrutura. Quando se discute software, independência de plataforma significa que um programa pode ser executado em qualquer computador.

Precisão	Refere-se ao nível de detalhe da informação existente num SIG. Informações de localização de alta precisão podem chegar ao centímetro, atributos de alta precisão especificam as características destes com grande detalhe. No entanto, os dados muito precisos podem ser incorrectos porque, por exemplo, os topógrafos podem cometer erros ou os dados podem ser introduzidos incorrectamente na base de dados. Assim deve-se distinguir entre a precisão e a correcção dos dados (Panagopoulos, 1999).
Processo	A palavra "processo", deriva do latim proceder e (processu) significando etimologicamente, "acto de proceder, de ir por diante; seguimento, curso, marcha".
Protocolo	Conjunto de regras sintácticas e semânticas que determinam o comportamento das entidades que interagem, incluindo a sequência, temporização, formato, e controlo dos fluxos e erros. Por exemplo, a Internet usa o protocolo de comunicações TCP/IP (Panagopoulos, 1999).
Protocolo de Transferência de Hipertexto	Protocolo de comunicação (na camada de aplicação segundo o Modelo OSI) que permite a transmissão de documentos de hipertexto por meio da rede.
Qualidade dos dados	Atributo de um conjunto de dados que define a sua adequação para um fim em particular, por exemplo a sua plenitude, exactidão espacial, actualidade, estrutura lógica, etc.
Resolução	Distância mínima que pode ser registada num sistema de medidas. Se um mapa tem uma resolução de 10 metros, então esse mesmo mapa não pode representar com exactidão elementos geográficos inferiores a 10 metros (Panagopoulos, 1999).
Serviço	Parte distinta da funcionalidade que é fornecida por uma entidade através de interfaces.
Simple Object Access Protocol (SOAP)	É um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída, utilizando tecnologias baseadas em XML.
Sistema Aberto	Sistema que implementa uma interface aberta de modo a promover a portabilidade das aplicações, a interoperabilidade, a diversidade, a gestão, a extensibilidade e a compatibilidade com sistemas mais antigos (Panagopoulos, 1999).
Sistema de Coordenadas Geográficas	Sistema de coordenadas numéricas usado para localizar e registar posições físicas específicas na superfície da Terra (Panagopoulos, 1999).
Sistema de Coordenadas Locais	Sistema criado para pequenas áreas geográficas. Os sistemas de coordenadas locais não podem ser expandidos para inclusão de áreas mais vastas sem perda de exactidão. Normalmente estes sistemas, são coordenadas do tipo (X, Y) em vez de sistemas de projecção (Panagopoulos, 1999).

Sistema de Informação Geográfica (SIG)	Não existe um consenso relativamente a este termo, contudo uma definição aproximada seria: Sistema para capturar, armazenar, verificar, integrar, manipular, analisar e visualizar dados georreferenciados. Isto normalmente envolve uma base de dados espacialmente referenciada e software apropriado. Um sistema SIG contém subsistemas para entrada de dados; representação, armazenamento e pesquisa de dados; gestão, transformação e análise de dados; geração de relatórios, gráficos e estatísticas; e saída de dados (Panagopoulos, 1999).
Sistema Global de Posicionamento (GPS)	Tecnologia de posicionamento por satélite que, através de correcção diferencial, assegura uma elevada precisão da localização. O posicionamento é obtido a partir do valor correspondente ao tempo gasto na transmissão de sinais desde pelo menos três satélites GPS, em orbita conhecida, até um receptor à superfície da Terra. Foi concebido inicialmente para fins militares, mas é actualmente muito utilizado em actividades na natureza. Para além de darem coordenadas permitem igualmente indicar rumos a seguir, altitude a que o seu utilizador se encontra, entre outras funções.
Sistema Operativo	Software que administra a execução dos programas e que pode proporcionar serviços tais como atribuição de recursos, encadeamento de tarefas, controlo das entradas e saídas, e gestão de dados (Panagopoulos, 1999).
Structured Query Language (SQL)	É uma linguagem padrão de programação interactiva que permite seleccionar, inserir e actualizar dados de uma base de dados. Embora o SQL seja padrão ANSI e ISO, muitos produtos de bases de dados suportam o SQL com extensões proprietárias.
Subespaço topológico	Em topologia, um subespaço topológico de um espaço X é um subconjunto de X munido da topologia relativa.
Tagged Image File Format (TIFF)	É um formato para ficheiros de imagem muito flexível, tipicamente utilizado por programas que controlam scanners (digitalizadores) ou câmaras digitais e por programas de manipulação de imagem. A flexibilidade advém do facto de o formato se basear no uso de tuplas (nome, valor), permitindo assim a um programa ignorar as que não reconhece (Panagopoulos, 1999).
Topologia	Refere-se a propriedades de formas geométricas que permanecem invariantes quando as formas são deformadas ou transformadas via flexão (bending), alongamento (stretching) ou contracção (shrinking). No âmbito dos SIG é a relação espacial que define a localização de fenómenos geográficos, uns relativamente aos outros, mas independentemente da distância ou direcção (Panagopoulos, 1999).
Uniform Resource Locator (URL)	Um URL (de Uniform Resource Locator), em português Localizador de Recursos Universal, é o endereço de um recurso (um arquivo, uma impressora etc.), disponível em uma rede; seja a Internet, ou uma rede corporativa ou uma intranet. Um URL tem a seguinte estrutura: protocolo://servidor/caminho/recurso

Universal Transverse Mercator (UTM)	Sistema Geo-Cartográfico que utiliza a chamada Representação Cartográfica de Mercator Transversa, a qual estabelece uma relação biunívoca entre pontos do Elipsóide Internacional de Hayford e pontos do plano. Este sistema de coordenadas é um sistema universal, pois subdivide o globo terrestre em 60 sectores (fusos) de 6 graus de amplitude e Portugal Continental encontra-se localizado na zona 29 com o meridiano central (deformação linear nula) a 9 graus oeste de Greenwich (Panagopoulos, 1999).

Lista de abreviações

1D	Uma dimensão
2D	Duas dimensões
3D	Três dimensões
4D	Quatro dimensões
API	Application Programming Interface
CDATA	XML Character Data
CDG	Conjunto de Dados Geográficos
CRS	Coordinate Reference System
CS	Coordinate System
DCP	Distributed Computing Platform
DTD	Document Type Definition
EPSG	European Petroleum Survey Group
GIF	Graphics Interchange Format
GIS	Geographic Information System
GML	Geography Markup Language
GPL	General Public License do GNU
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ITRF	International Terrestrial Reference Frame
ITRS	IERS Terrestrial Reference System
JNLP	Java Network Launching Protocol
JPEG	Joint Photographic Experts Group
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KVP	Keyword Value Pair
LGPL	Lesser General Public License do GNU
MDB	Message Driven Beans
MIME	Multipurpose Internet Mail Extensions
NAD	North American Datum
ODBC	Open Database Connectivity
OGC	The Open Geospatial Consortium, Inc.
OWS	OGC Web Service
PNG	Portable Network Graphics
RDF	Resource Description Framework
RFC	Request for Comments
RGB	Red-Green-Blue
SIG	Denominação em português da abreviatura GIS
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
UCUM	Unified Code for Units of Measure
URL	Uniform Resource Locator
URN	Universal Resource Name
WCS	Web Coverage Service
WebCGM	Web Computer Graphics Metafile
WFS	Web Feature Service
WGS	World Geodetic System
WMS	Web Map Service

WPS	Web Processing Service
XML	Extensible Markup Language

Resumo

Esta dissertação apresenta uma arquitectura interoperável que permite lidar com a obtenção, manipulação, processamento e análise de informação geográfica. A aplicação 3D, implementada como parte da arquitectura, para além de permitir a visualização e manipulação de dados dentro de um ambiente 3D, oferece métodos que permitem descobrir, aceder e usar geo-processos, disponíveis através de serviços Web. A interacção com o utilizador é também feita através uma abordagem que quebra a típica complexidade que a maioria dos Sistemas de Informação Geográfica apresenta. O recurso à programação visual reduz a complexidade do sistema, e permite aos operadores tirar proveito da localização e de uma abstracção de um processo complexo, onde as unidades de processamento são representadas no terreno através de componentes 3D que podem ser directamente manipuladas e ligadas de modo a criar encandeamentos complexos de processos. Estes processos podem também ser criados visualmente e disponibilizados online.

INTEROPERABILITY, PROCESSING AND VISUAL ANALYSIS OF GEOPATIAL DATA

This thesis presents an interoperable architecture mainly designed for manipulation, processing and geographical information analysis. The three-dimensional interface, implemented as part of the architecture, besides allowing the visualization and manipulation of spatial data within a 3D environment, offers methods for discovering, accessing and using geo-processes, available through Web Services. Furthermore, the user interaction is done through an approach that breaks the typical complexity of most Geographic Information Systems. This simplicity is in general archived through a visual programming approach that allows operators to take advantage of location, and use processes through abstract representations. Thus, processing units are represented on the terrain through 3D components, which can be directly manipulated and linked to create complex process chains. New processes can also be visually created and deployed online.

Capítulo I: Introdução

Numa das obras de Jorge Luís Borges (1899-1986), intitulada "O Aleph", vem o conto "Os dois reis e os dois labirintos". Diz o conto:

" (...) nos primeiros dias houve um rei das ilhas da Babilónia que reuniu os seus arquitectos e magos e lhes mandou construir um labirinto tão complexo e subtil que os varões mais prudentes não se aventuravam a entrar nele, e os que nele entravam se perdiam. Essa obra era um escândalo, pois a confusão e a maravilha são atitudes próprias de Deus e não dos homens. Com o correr do tempo, chegou à corte um rei dos Árabes, e o rei da Babilónia (para zombar da simplicidade do seu hóspede) fez com que ele penetrasse no labirinto, onde vagueou humilhado e confuso até ao fim da tarde. Implorou então o socorro divino e encontrou a saída. Os seus lábios não pronunciaram queixa alguma, mas disse ao rei da Babilónia que tinha na Arábia um labirinto melhor e que, se Deus quisesse, lho daria a conhecer algum dia. Depois regressou à Arábia, juntou os seus capitães e alcaides e arrasou os reinos da Babilónia com tão venturosa fortuna que derrubou os seus castelos, dizimou os seus homens e fez cativo o próprio rei. Amarrou-o sobre um camelo veloz e levou-o para o deserto. Cavalgaram três dias, e disse-lhe: "Oh, rei do tempo e substância e símbolo do século, na Babilónia quiseste-me perder num labirinto de bronze com muitas escadas, portas e muros; agora o Poderoso achou por bem que eu te mostre o meu, onde não há escadas a subir, nem portas a forçar, nem cansativas galerias a percorrer, nem muros que te impeçam os passos". Depois, desatou-lhe as cordas e abandonou-o no meio do deserto, onde morreu de fome e de sede (...) "

(Borges, 1998)

Nas imagens antagónicas do deserto e do labirinto é possível encontrar a relação entre o excesso de redundância do deserto e o excesso de informação do labirinto. Hoje, observa-se a passagem de uma fase de ausência de dados para uma nova fase, em que grandes quantidades de dados não significam, exactamente, um ganho de informação. Passa-se da dificuldade em obter dados, para a dificuldade em organizar e processar grandes conjuntos de dados.

Num discurso proferido na Califórnia Science Center, em 31 de Janeiro de 1998, Al Gore diz:

> "A new wave of technological innovation is allowing us to capture, store, process and display an unprecedented amount of information about our planet and a wide variety of environmental and cultural phenomena. Much of this information will be "georeferenced" - that is, it will refer to some specific place on the Earth's surface. The hard part of taking advantage of this flood of geospatial information will be making sense of it - turning raw data into understandable information. Today, we often find that we have more information than we know what to do with"
>
> (Gore, 1998)

A ideia é organizar o potencial dos dados já existentes e torná-los acessíveis à comunidade mundial, com o objectivo de criar uma espécie de laboratório "sem paredes". Este aspecto é também salientado no seu discurso:

> "We have an unparalleled opportunity to turn a flood of raw data into understandable information about our society and out planet. This data will include not only high-resolution satellite imagery of the planet, digital maps, and economic, social, and demographic information. If we are successful, it will have broad societal and commercial benefits in areas such as education, decision-making for a sustainable future, land-use planning, agricultural, and crisis management. The Digital Earth project could allow us to respond to manmade or natural disasters - or to collaborate on the long-term environmental challenges we face"
>
> (Gore, 1998)

Uma quantidade de dados nunca antes disponível está-se a tornar acessível. A questão é: sem os devidos cuidados com processos metodológicos para a manuseio e a exploração dos dados, pode cair-se na situação do labirinto, pois o excesso de informação é tão dramático como a falta de informação (Moura, 2000). O excesso de informação (o labirinto), se não é enfrentado de um modo correcto, pode levar a conclusões sustentadas, ou uma situação onde não é possível encontrar a informação desejada (a ausência, o deserto).

1.1. Inexistência de interoperabilidade

Desde há muitos anos, que diversas organizações e companhias têm vindo a disponibilizar serviços que permitem aceder a informação geográfica. Esta informação é tipicamente guardada por parte dessas entidades, em bases de dados, ficheiros de texto ou Sistemas Geográficos de Informação (SGI).

Devido ao facto de não existir um padrão que seja aceite e usado por todos na representação de dados georreferenciados, e devido ao facto dos recursos geográficos serem desenhados com diversas finalidades, estes sistemas são tradicionalmente implementados de acordo com formatos e protocolos diferentes, que em muitos casos são também proprietários (De Amicis et al., 2008). O caso ainda se agrava mais quando os fornecedores entregam sistemas fechados (Fallman, 2004), levando a que muitos consumidores fiquem limitados apenas a uma solução, constituída por uma aplicação cliente capaz de interagir apenas com algumas implementações específicas. Este tipo de limitação leva à criação de uma espécie de mundo formado por pequenas ilhas de informação. Contudo, é possível observar a nível internacional, uma necessidade crescente de especificações de interface, publicamente documentadas, aceites como padrão e implementadas pelo maior número possível de fornecedores (Open Geospatial Consortium, 2008), de modo a que a interoperabilidade entre os diferentes produtos possa ser assegurada.

Como resposta a esta necessidade, actualmente já começam a emergir alguns padrões que permitem a existência de interoperabilidade entre as diversas infra-estruturas geográficas baseadas em serviços web, algo impensável há pouco tempo atrás. Os esforços de padronização mais notáveis estão a ser levados a cabo pelo Open Geospatial Consortium (OGC), ISO - International Organization for Standardization (ISO, 2008) e pelo CEN - European Committee for Standardization (CEN, 2008), através da oferta de padrões abertos de computação para cartografia e geoprocessamento.

Estes esforços de harmonização e padronização receberam também um grande impulso com o aumento de visualizadores 3D para consumidores, como por exemplo, o Google Earth (Google Earth, 2008), Microsoft Virtual Earth (Microsoft Virtual Earth, 2008) e NASA World Wind (NASA World Wind, 2008). O sucesso destas aplicações deve-se em parte ao facto de terem uma capacidade de visualizar dados

geográficos dentro de um ambiente 3D (De Amicis et al., 2008), e de conseguirem ocultar os detalhes técnicos relativos à procura, acesso e obtenção da informação.

1.2. Inexistência de conformidade nos dados

Como consequência dos primeiros passos da interoperabilidade e das vantagens que esta pode trazer, o número de pessoas e instituições que partilham informação geográfica está a colidir com o aparecimento de serviços espaciais web distribuídos (WS), capazes de fornecer um acesso segundo padrões a dados geográficos, e provando que Boucelma (Boucelma, Lacroix, & Essid, 2002) estava certo quando afirmava que a criação de serviços espaciais web distribuídos era um conceito chave para a interoperabilidade à escala global.

Contudo a interoperabilidade acarreta novas dificuldades. Segundo estudos da União Europeia (EU), 52% da informação no sector público na Europa é constituída por informação geoespacial (FTK - Research Institute for Telecommunication, 2008) geralmente caracterizada pela sua alta duplicação nos dados (Ordnance Survey, 2006). Isto deve-se em grande parte ao facto dos dados serem guardados com diferentes estruturas e classificações (por exemplo de acordo com um nome técnico, uma abreviatura, um nome popular, etc.), e demonstra também que qualquer desenvolvimento tecnológico que facilita o uso ou reutiliza informação geoespacial pode trazer benefícios quer sociais quer económicos, a nível nacional ou internacional (Conti, De Amicis, Piffer, & Simões, 2008). Estas vantagens advêm do facto, de por exemplo, apenas se despender recursos para obter a informação uma vez.

Ao nível internacional este problema está também a originar várias iniciativas com o objectivo de promover a definição de infra-estruturas partilhadas baseada em Padrões Internacionais. Entre essas iniciativas as mais relevantes são GMES (Global Monitoring for Environment and Security, 2008), GEOSS (Global Earth Observation System of Systems, 2008) e a European Spatial Data Infrastructure (ESDI). Também a directiva europeia INSPIRE (INfrastructure for SPatial InfoRmation in Europe) (INSPIRE, 2008), que entrou em força em Março de 2007, tem sido uma das principais propulsoras nestas iniciativas. A INSPIRE define as fundações de uma Framework tanto legal como técnica, com a intenção de fornecer um acesso harmonizado a dados ambientais de interesse pela Europa. Esta directiva representa um passo importante a nível Europeu, que irá conduzir a um grande impacto a nível tecnológico, operacional e político, quer dentro de administrações locais, regionais ou nacionais (Conti, De Amicis, Piffer, & Simões, 2008).

1.3. Importância de uma Analítica Geo-Visual

O aparecimento de padrões tem como consequência que uma grande quantidade de informação esteja acessível de um modo interoperável. Adicionalmente, e com a heterogeneidade e multidimensionalidade dos dados, a sua visualização através de tabelas ou formulários começa a ser lenta e obsoleta.

É necessária uma nova abordagem para este tipo de dados, de modo a tirar proveito da sua natureza georreferencial. É também necessária a visualização destes sistemas de modo a que maximize o número e a complexidade das ideias, e ofereça uma compreensão profunda da respectiva informação, pois o ser humano começa também a perder a sua capacidade de análise com o crescimento e complexidade dos dados. São necessários novos métodos que permitam visualizar, reconhecer padrões e obter ideias em

tempos de pesquisa reduzidos, tentando explorar ao máximo as suas fontes de input mais eficazes como a visão e audição.

1.4. Contribuições desta tese

Esta tese descreve uma Arquitectura Serviço-Orientada (SOA) que promove a interoperabilidade de dados através do uso de serviços web, que podem ser disseminados pelo consumidor através de uma aplicação 3D. Ainda como parte da infra-estrutura SOA, é apresentada uma solução completa para uma plataforma cliente-servidor, onde são tidos em conta mecanismos com a tolerância a erros, transacções assíncronas e independência de plataforma. É de acentuar que mecanismos que permitam transacções assíncronas de modo eficaz são praticamente inexistentes nas aplicações existentes que tentam promover a interoperabilidade.

A aplicação 3D disponibilizada providencia uma ambiente 3D que de um modo simples e intuitivo permite a visualização, edição, processamento e análise dos dados. Este protótipo é extremamente pequeno, pode ser executado da web, e efectua todo o processamento de dados através de serviços web, de modo a que qualquer pessoa que use um computador normal tenha a possibilidade de efectuar o seu trabalho, mesmo que este exija uma grande capacidade de processamento.

Além disso, o protótipo possibilita também a programação visual de processos complexos de um modo inovativo. A programação é efectuada através do uso de expressões visuais, arranjos espaciais de texto e símbolos gráficos, usados como elementos de sintaxe ou de anotação secundária, e que podem ser manipulados pelo utilizador de um modo interactivo com a vista a tornar as tarefas mais fáceis, rápidas de construir e de analisar. Estes processos complexos podem ser programados, entendidos ou obtidos de serviços online de processamento em run-time, e serem executados remotamente.

Este protótipo tem como objectivo tornar acessível a qualquer pessoa o uso de funcionalidades que anteriormente estavam limitadas as especialistas em SIG ou que exigiam muito tempo de aprendizagem, através de uma interface interactiva e intuitiva.

1.5. Estrutura da tese

No próximo capítulo é apresentado o estado actual dos sistemas SIG e da Analítica Visual em sistemas SIG. São introduzidos os diversos protocolos usados, e é apresentada uma breve visão sobre o software Open Source SIG existente assim como as suas licenças e implicações.

O capítulo III apresenta as ferramentas usadas, são descritas algumas alterações que devem ser aplicadas aos diversos protocolos usados de modo a que estes sejam mais flexíveis e eficazes. É feita uma descrição detalhada sobre a arquitectura e detalhes técnicos da implementação. Posteriormente, no capítulo seguinte é apresentado um caso de estudo e uma análise dos resultados obtidos.

Por fim a tese termina com uma pequena secção dedicada a conclusões e perspectivas futuras.

Capítulo II: State-of-Art

2.1. Objectivos do capítulo II

Este capítulo tem com objectivo apresentar o estado actual de bibliotecas e ferramentas SIG e intruduzir as especificações que são usadas.

O capítulo inicia-se com a apresentação do estado actual da analítica geo-visual e dos sistemas SIG. Depois são apresentadas algumas das especificações que são usadas. Estas especificações fazem em muitos dos casos uso da GML e descrevem a sintaxe que deve ser usada de modo a que as diversas componentes da arquitectura possam comunicar entre si.

É feita uma pequena introdução à GML, que permite responder a questões como: o que é, para que serve e como pode ser optimizada. Posteriormente, são comparadas algumas tecnologias de binding XML de modo a explicar o porquê da escolha do JAXB para o protótipo.

Para finalizar, é apresentada uma breve visão sobre o state-of-art do software open-source SIG existente assim como as suas licenças e implicações. Esta subsecção tem como objectivo justificar o uso de algumas bibliotecas usadas na implementação e contrastar a aplicação implementada com outras aplicações Free and Open Source Software (FOSS) existentes.

2.2. Analítica Geo-Visual

Uma computação gráfica interactiva pode ser extremamente efectiva com uma vasta gama de informações de n-dimensões. Com a abordagem correcta em termos de interface e itens de informação, grandes quantidades de dados podem ser facilmente compreendidas por um observador humano através do uso de gráficos 3D (Wright, 1997). É de facto um dado adquirido que a visualização gráfica fornece uma ajuda mental adicional, que aumenta as habilidades cognitivas (Card, Mackinlay, & Shneiderman, 1999; De Amicis et al., 2008). Quando a informação é apresentada visualmente, novas capacidades humanas inatas podem ser usadas para perceber e processar os dados. As técnicas de visualização de informação ampliam a cognição através do aumento de recursos mentais, reduzindo os tempos de procura, melhorando o reconhecimento de padrões, a inferência de conclusões e aumentando do campo de monitorização (Card, Mackinlay, & Shneiderman, 1999).

Por esta razão nos últimos anos a comunidade de pesquisa tem se focado numa nova disciplina chamada Analítica Visual. A Analítica Visual surgiu com o objectivo de promover formas efectivas de aceder a grandes quantidades de dados multidimensionais, e de estudar novos métodos que permitam filtragens rápidas de informação complexa e identificação de padrões importantes. No que diz respeito a este tópico, a agência americana National Visualization and Analytics Center (NVAC) tem tido um dos papéis principais, dando uso esta disciplina em contextos como ambiente e segurança nacional. Uma análise visual de dados oferece vantagens a analistas porque providencia sugestões visuais que os podem ajudar a formular um conjunto de modelos viáveis. As fraquezas desta abordagem é que há frequentemente possibilidades infinitas em termos de cartografias e visões, e há um elevado potencial para que a informação sobrecarregue em campos de informação densos (National Visualization and Analytics Center, 2004).

Também e segundo Hetzler e Turner (Hetzler & Turner, 2004), muitos dos sistemas de analise visual existentes são data-centricos, e focam-se num tipo particular de dados para os quais providenciam ambientes separados, que podem ser combinados para analisar os diferentes tipos de informação, contudo geralmente cada um deles enfatiza sobre os outros. Por esta razão vários autores (Andrienko & Andrienko, 2004) têm explorado como a síntese de informação pode permitir ao analista controlar informação dinâmica de todos os tipos em um ambiente sem barreiras (Conti et al., 2008).

Um ambiente ideal de análise deveria ter uma integração sem restrições da parte computacional e das técnicas visuais. Por exemplo, a avaliação global visual pode ser baseada em algumas transformações preliminares apropriadas aos dados e a tarefa. Focalização interactiva, selecção e filtragem podem ser usados para isolar dados associados com uma hipótese e passados então a um motor de análise. Os resultados podem ser sobrepostos na informação original de modo a sublinhar a diferença entre os dados originais e os dados processados através de um modelo de computação, com os erros realçados visualmente. Este processo poderia ser iterado se o modelo resultante não correspondesse à precisão desejada dos dados, ou devido ao facto do analista re-focalizar um subespaço de informação diferente (De Amicis et al., 2008; National Visualization and Analytics Center, 2004).

Outro problema que pode ser encontrado é que muitas das técnicas de visualização para análises de interacção de eventos complexos, apenas têm a capacidade de apresentar informações sobre uma única dimensão, sendo senso comum que estes tipos de visualização são bem compreendidos (Kapler & Wright, 2004). Actualmente, alguns sistemas são capazes de usar animações para apresentar a dimensão tempo. O tempo pode eventualmente ser controlado através de um scroll, onde a informação espacial ou outros tipos de apresentações variam para reflectir o estado da informação nesse momento. Contudo esta técnica assenta sobre a memória humana a curto prazo para reter as alterações temporais e padrões. Numa tentativa de melhorar este problema Tufte usa uma técnica semelhante a uma fita de filme onde cada frame representa as condições ou gráficos para cada momento de tempo capturado de um modo incremental (Tufte, 1990). Estas imagens podem ser interpretadas separadamente ou através de comparações lado a lado de modo a detectar as diferenças. Esta técnica é muito dispendiosa em termos de espaço visual devido ao facto de que uma imagem deve ser gerada para cada momento de interesse, o que pode ser problemático quando se tenta apresentar ao mesmo tempo múltiplas imagens num tamanho adequado. Baseadas nesta técnica existe outras, como por exemplo a ilustrada na Figura 1, que tentam apresentar as diversas imagens de um modo mais organizado.

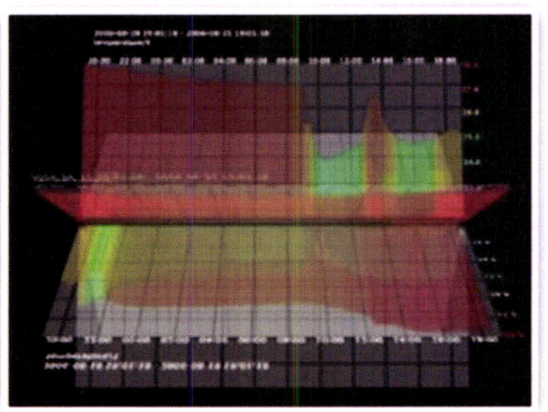

Figura 1 - Diagrama Rotacional (Einsfeld, Ebert, & Wölle, 2007)

Uma técnica adicional a mencionar é o uso de múltiplas views ligadas de modo a suportar análises multi-variadas (ver Figura 2). Por exemplo incluindo series de analises de dados temporais numa view e um mapa noutra view (Eick & Wills, 1995). Um relacionamento interactivo dos dados seleccionados através dos múltiplos views melhora a técnica de Tufte (Tufte, 1990).

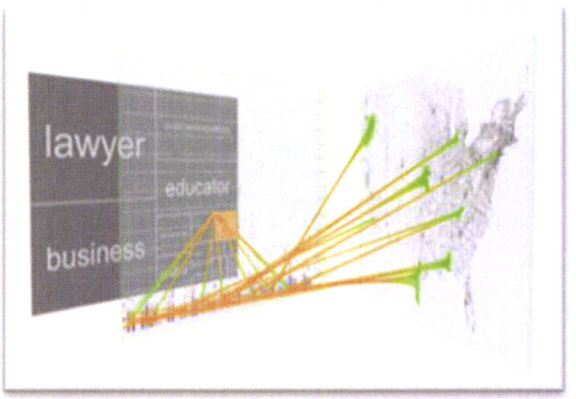

Figura 2 – VisLink (Collins & Carpendale, 2007)

Kapler et al. sugerem posteriormente uma abordagem de certo modo radical para a integração de espaço-tempo chamada GeoTime (Kapler & Wright, 2004; Eccles, Kapler, Harper, & Wright, 2008). GeoTime (ver Figura 3) apresenta um ambiente que liga fortemente a transformação de dados e a sua visualização, o que resulta num um processo de análise mais poderoso, pois permite ao utilizador juntar as vantagens de cada método.

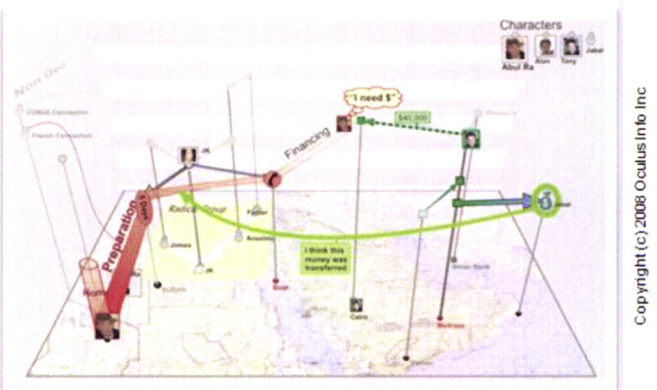

Figura 3 – Stories in GeoTime (Eccles, Kapler, Harper, & Wright, 2008)

Actualmente, existem quatro abordagens gerais por construir um software de visualização (National Visualization and Analytics Center, 2004).

- Construção de uma ferramenta de visualização de uso general que tem como objectivo um domínio particular. Exemplos de alguns sistemas que seguem esta abordagem são OpenDx, IN-SPIRE, XmdvTool, Spotfire, Rivet e Prefuse.
- Construção de bibliotecas de visualização baseadas em componentes como InfoVisToolkit, ILOG Visualization ou Vz Visualization Library.
- Construção de componentes de visualização que trabalham bem em conjunto. Exemplos desta abordagem incluem as componentes de visualização North's Snap-Together Visualizations e Eick's ADVIZOR.
- Sistemas que automaticamente geram software de visualização. Estes sistemas seguem técnicas baseadas em regras para determinar segundo a tarefa e o tipo de dados, o tipo de visualização apropriado.

2.3. Aplicações SIG 3D/4D

Com a capacidade de processamento gráfico que os computadores actuais possuem, e com o número elevado de novas especificações e protocolos definidos pelo OGC, começa-se a notar uma tendência cada vez maior na criação e adopção de geoserviços 3D, essenciais para ambientes de gestão de tarefas avançadas como ilustrado por Zipf (Zipf, 2008). Alguns exemplos destas tendências já estão publicamente disponíveis, como o caso do Heidelberg 3D (GDI3D, 2008), CityServer3D (Haist & Coords, 2005) e Berlin 3D (Berlin 3D, 2008).

As aplicações SIG em 3D podem oferecer grande usabilidade e interactividade que facilita a troca e disseminação de informação espacial entre as diversas entidades (Grasso, Cervone, Singh, & Kafatos, 2006; Grasso & Singh, 2008) e proporciona uma visualização 3D que pode fornecer uma ajuda fundamental para a percepção das relações espaciais 3D entre a informação e os conjuntos de dados (Conti, De Amicis, Piffer, & Simões, 2008). Contudo, e como referido por Sliuzas (Sliuzas, 2003) a complexidade da maioria dos sistemas SIG é ainda de longe demasiado complexa para pessoas como gestores ou administradores que geralmente tem pouco conhecimento em SIG.

Actualmente, pode ser encontrada uma literatura diversa sobre os princípios gerais da navegação e interacção 3D. Thorndyke e Hayes-Roth, assim como muitos outros autores (Hunt & Waller, 1999; Robinett & Holloway, 1992; Siegel & White; Tan, Robertson, & Czerwinski, 2001; Witmer, Bailey, Knerr, & Parsons, 2002; Thorndyke & Hayes-Roth, 1982), estudaram as diferenças no conhecimento espacial adquirido através de mapas e exploração. Darken e outros exploraram a cognição e os princípios de design e como podem ser aplicados a mundos virtuais de grandes dimensões (Darken & Sibert, Navigating in Large Virtual Worlds, 1996; Robinett & Holloway, 1992; Vinson, 1999; Tan, Robertson, & Czerwinski, 2001; Darken & Sibert, A Toolset for Navigation in Virtual Environments, 1993). Furnas explorou transversalidade da visão e a navegabilidade em grandes estruturas de dados (Furnas, 1997; Tan, Robertson, & Czerwinski, 2001). Em 1995, Koller et al. (Koller, D. et al, 1995) criaram um ambiente 3D que funciona em tempo real, especificamente desenhado para trabalhar com dados geográficos. Shumilov et al. (Shumilov, Thomsen, Cremers, & Koos, 2002) apresentaram um protótipo para uma aplicação SIG que permite a construção de modelos 3D e 4D através de geometrias VRML dependentes do tempo, usadas para representar as variações temporais. Em 2002, Zlatanova apresentou uma pesquisa sobre o software SIG mais usado, e onde foram revistos vários sistemas como ArcGIS (ESRI, 2008), Imagine VirtualGIS (ERDAS, 2008), PAMAP GIS Topographer (PCIGEOMATICS, 2008) e GeoMedia Terrain (Intergraph, 2007). Zlatanova concluiu nessa pesquisa que alguns passos iniciais foram dados em termos de visualização e animação de dados e objectos espaciais 3D, mas que essas aplicações 3D não apresentavam geralmente as funções básicas de SIG (Brooks & Whalley, 2005) como a pesquisa (ou era geralmente efectuada através de interacções baseadas em menus), a criação, a manipulação e análise

de geo-objectos 3D (Darken & Sibert, 1996) em 3D. Mais recentemente, Brooks e Whalley (Brooks & Whalley, 2005) desenvolveram um sistema capaz de integrar camadas de dados (tradicionalmente estruturas de dados adoptadas dentro dos sistemas SIG), dentro de uma vista 3D. Clark e Maurer (Clark & Maurer, 2006) apresentaram um sistema SIG integrado e interactivo que prova que os SIG 3D podem ser estendidos de modo a suportar funcionalidades como a manipulação directa de características do terreno e a criação de conteúdo dinâmico. O sistema, desenvolvido como aplicação autónoma e monolítica, tem sua limitação principal no facto que requer uma ferramenta externa para construir o conteúdo. De um modo semelhante, Ohigashi et al. (Ohigashi, Guo, & Tanaka, 2006) apresentaram uma aplicação 3D ligada aos habituais SIG 2D através de "shandow copies".

Outros exemplos de pesquisa em protótipos 3D que existem incluem Terrafly (Rishe, Sun, Chekmasov, Andriy, & Graham, 2004), GeoVR (Huang & Lin, 1999) e TerraVisionII (Reddy, Leclerc, Iverson, & Bletter, 1999). Um sistema notável, chamado GeoTime (Kapler & Wright, GeoTime Information Visualization, 2004) propõe uma solução para o problema de integrar eventos temporais em SIG interactivos. O modelo de planeamento SIG Adelaide 3D (GISCA, 2008) é um exemplo de um SIG 3D que providencia visualizações 3D de dados sociais e ambientais dentro de uma vista 3D da cidade (Brooks & Whalley, 2008). Duas outras aplicações também interessantes são GeoZui3D (Ware, Plumlee, Arsenault, Mayer, & Smith, 2001) e VGIS (Köller, Lindstrom, Ribarsky, Hodges, Faust, & Turner, 1995). GeoZui3D é um SIG marinho que suporta entradas de dados em tempo real. VGIS é um sistema global integrado SIG e um sistema visual de Brooks e Whalley (Brooks & Whalley, 2008). Contudo nenhum dos trabalhos atrás mencionado consegue suportar a troca de dados entre múltiplas instâncias da aplicação de modo a suportar actividades colaborativas e cooperativas (Conti, De Amicis, Piffer, & Simões, 2008).

O suporte para CSCW (Computer Supported Collaborative Work) tem vindo a tornar-se um factor de grande importância, e como consequência disso, vários autores desenvolveram técnicas e tecnologias diferentes para suportar a colaboração dentro de ambientes SIG 3D. Os primeiros esforços à volta destas análises colaborativas de dados espaciais em ambientes 3D foram apresentados por Manoharan et al. (Manoharan, Taylor, & Gardiner, 2002), que apresentam um ambiente de trabalho colaborativo, interactivo e independente da localização. Dentro deste mesmo contexto, Ganesan et al. (Ganesan, Estrin, & Heidemann, 2003) ilustram a importância e viabilidade da integração de dados sensoriais e redes de sensores wireless dentro de um sistema distribuído, mantendo a relação temporal e alcançando assim um sistema distribuído 4D-enabled. Outros trabalhos como o de MacEachren (MacEachren, 2005) estenderam o conceito de geovisualização de modo a suportar actividades colaborativas de utilizadores que cooperam dentro de um sistema de redes. Em pesquisas recentes, mais esforços no mesmo problema são apresentados por Haist e Korte (Haist & Korte, Adaptive streaming of 3d gis geometries and textures for interactive visualisation of 3d city models, 2006), que propõem um sistema cliente-servidor capaz de assegurar a troca de conteúdo 3D através de uma rede pública (Conti, De Amicis, Piffer, & Simões, 2008).

No que diz respeito a aplicações comerciais, é possível encontrar um grande número delas em uso pela comunidade técnica de SIG, e que providenciam algumas funcionalidades 3D (Environment System Research Institute, 2008), contudo muitas delas são principalmente de visualização, e muito poucas estão ligadas ao geoprocessamento. Também nos últimos anos tem aumentado o aparecimento de aplicações para o consumidor como o Google Earth (Google Earth, 2008), Microsoft Virtual Earth (Microsoft Virtual Earth, 2008) ou NASA World Wind (NASA World Wind, 2008). Estas aplicações, geralmente referidas como geobrowsers 3D, apresentam uma grande usabilidade, o que tem contribuído para o seu uso amplo pelo público em geral. A sua adopção global levou também à abertura do campo geomático, tradicionalmente limitado a especialistas do domínio, a um público mais amplo que agora pode usufruir da tendência socio-tecnológica conhecida como "Neogeografia" (Conti, De Amicis, Piffer, & Simões, 2008; Turner, 2006).

Várias experiências (Coughlin, Cuff, & Krause, 2008) têm demonstrado que geobrowsers 3D podem providenciar uma contribuição significante a especialistas assim como também a outros operadores. O caso de desastres naturais como o furação Katrina e o terramoto de 2005 no Paquistão demonstrada a importância que estes geobrowsers 3D podem ter na ajuda a agências e equipas de resgate durante grandes crises ambientais (Conti, De Amicis, Piffer, & Simões, 2008; Nourbakhsh, Sargent, Wright, Cramer, McClendon, & Jones, 2006).

A rápida evolução de tecnologias da observação da terra, satélites e sensores está a criar novos desafios no desenvolvimento dos geobrowsers 3D. O problema advém do facto destas tecnologias estarem a gerar um crescimento constante de dados, que necessitam de novos métodos de processamento, análise e redução de complexidade. Um exemplo claro disso é a NASA Earth Science Data and Information System (ESDIS, 2008) que hoje já conta com vários Petabytes (1000 Gigabytes) de dados, reportando em 2005 um crescimento a uma taxa de 3.5 Terabytes por dia. Outro exemplo notável é a missão do satélite ESA Envisat que está a produzir 160 Gigabytes de dados por cada orbita efectuada à terra, podendo efectuar até 14 orbitas por dia. O que dá 1 Petabyte a cada 5 anos. Também a Microsoft anunciou em Abril de 2008 o seu novo data center em Boulder (USA) capaz de armazenar dados até 15 Petabytes disponíveis para a sua solução Microsoft Virtual Earth (Conti, De Amicis, Piffer, & Simões, 2008).

2.4. Open Geospatial Consortium (OGC)

Criado em 1994, o consórcio internacional Open Geospatial era anteriormente conhecido por Open GIS Consortium ou Consórcio OpenGIS. Actualmente o termo OpenGIS é apenas uma marca registada que faz referência às diversas especificações do consórcio.

O Open Geospatial Consortium é uma organização internacional voluntária de padrões de consenso, que conta com mais de 300 organizações comerciais, governamentais, não-lucrativas e diversas instituições de pesquisa distribuídas pelo mundo, que colaboram com o OGC num processo de consenso aberto, encorajando o desenvolvimento e a implementação de padrões para conteúdo e serviços geomáticos, de processamento e de troca de dados.

As especificações mais importantes do OGC, e que serão abordadas nesta arquitectura são:

- Simple Features Specification (SFS) – Define um formato para armazenamento, leitura, análise e actualização de unidades de dados geográficos.
- Geography Markup Language (GML) – Linguagem que permite o transporte e armazenamento de informações geográficas.
- OGC Web Services (OWS) – Especificações de serviços Web:
    - Web Map Service (WMS) – Serviço de mapas pela Internet.
    - Web Feature Service (WFS) – Serviço de características pela Internet.
    - Web Processing Service (WPS) – Serviços de processamento pela Internet.
- Symbology Encoding (SE) – Linguagem de tematização de uma característica.
- Styled Layer Descriptor (SLD) – Estende WMS para suportar tematização através da SE.
- Catalogue Service/Web (CSW) - Serviço de Catálogo pela Internet.

Das especificações apresentadas, pode-se considerar a SFS como a mais importante, já que a mesma define a organização dos dados espaciais numa Base de Dados Geográfica e as funções (análises) mais importantes de um SIG (análises topológicas, análises espaciais, etc.). Desta forma, é aconselhável que

uma instituição, ao usar uma solução, exija pelo menos que os dados vectoriais sejam armazenados de acordo com esta especificação (Uchoa & Ferreira, 2004).

O OGC tem também uma relação próxima com o ISO/TC 211 (Informação Geográfica/Geomática). A especificação abstracta tem também vindo a ser progressivamente substituída pela série ISO 19100, em desenvolvimento por este comité.

2.5. Simple Features Specification (SFS)

Esta especificação define um formato, de acordo com o SQL padrão para armazenamento, leitura, análise e actualização de "características simples" (dados geográficos) através de uma API (ODBC). O OGC define uma "característica simples" como uma composição de atributos espaciais e metadados. Estas características são baseadas em geometrias 2D com interpolação linear entre os vértices. O PostGIS é o módulo do PostgreSQL (SGBD de código aberto) que implementa essa especificação e estende as geometrias para 4 dimensões com inúmeras funcionalidades adicionais (Uchoa & Ferreira, 2004). O documento do OGC #06-104 (Herring, 2006) define todos os detalhes que essa interface deve respeitar.

2.6. Geographic Markup Language (GML)

Geography Markup Language (GML) é uma gramática XML definida pelo OGC que providencia objectos que tornam possível a descrição da geografia, nomeadamente, expressar características, topologia, unidades de medida, tempo, sistemas de referência de coordenas etc., assim como a modelação de sistemas geográficos, transacções de dados geográficos e armazenamento de informação espacial na Internet.

Porém, é necessário limitar a interpretação de dados de GML ao que é estritamente necessário para visualizar o conteúdo. Como a GML é um documento de XML, os mapas codificados em GML podem tornar se documentos de grandes dimensões.

O modelo GML era inicialmente baseado em RDF, mas subsequentemente, o OGC introduziu esquemas XML para conectar as várias bases de dados geográficos existentes. O resultado foi a GML 3.0, que possui também tipos primitivos como:

- Característica
- Geometria GML
- Referência de coordenadas
- Tempo
- Características dinâmicas
- Cobertura (incluindo imagens geográficas)
- Unidade de medida
- Estilo de apresentação de mapa

2.6.1. Definição de Perfil em GML

Um perfil é um conjunto de restrições à GML, que devem ser expressas através de um documento XML. A intenção do perfil é simplificar a adopção do GML, para facilitar a adopção do padrão.

Para expor os dados de uma aplicação geográfica com GML, uma comunidade ou organização cria um esquema XML específico da aplicação. Esse esquema descreve os tipos de objectos, e quais os dados que devem ser expostos pela aplicação (Universidade Federal do Estado do Rio de Janeiro, 2007). Usando esquemas XML faz com que seja também fácil estender o GML de modo a incluir metadados específicos de um fornecedor (Kwok & Nithianandan, 2006).

2.6.2. Definição de Característica geográfica

A GML define as características como distintas de objectos geométricos. Uma característica é um objecto de uma aplicação que representa uma entidade física, como um prédio, uma estrada, ou uma pessoa. Uma característica pode ou não ter aspectos geométricos. Um objecto geométrico define uma localização ou região além de uma entidade física. A distinção entre características e objectos geométricos em GML contrasta com modelos utilizados por Sistemas de Informação Geográficos (SIG), que não fazem tal distinção (Universidade Federal do Estado do Rio de Janeiro, 2007).

Em GML uma característica pode ter várias propriedades que descrevem aspectos geométricos da característica. GML também suporta a partilha das propriedades entre as características. As propriedades remotas são uma capacidade genérica adoptada da RDF. Assim, um atributo xlink:href numa propriedade geométrica da GML significa que o valor da propriedade é o recurso referenciado pelo link (Universidade Federal do Estado do Rio de Janeiro, 2007). Por exemplo, uma característica que representa um edifício num determinado esquema GML pode ter a sua posição especificada por um tipo primitivo da GML, como Point. No entanto, o edifício é uma entidade separada do Point, o qual apenas define a sua posição.

2.7. Análise gramatical da GML e técnicas de optimização

Actualmente existem diversas tecnologias que permitem fazer uma análise gramatical a documentos XML. Três das bibliotecas mais relevantes neste campo são o GML4J, XMLBeans e JAXB.

2.7.1. GML4J

GML4J é uma tentativa de aceder a dados em XML usando objectos concretos em Java, como Característica, Geometria, assim como os métodos que são específicos a estes objectos. Esta é uma tentativa de abstrair o acesso aos dados da GML através de elementos e nomes dos atributos.

GML4J usa com implementação uma abordagem que representa todo o documento XML numa árvore em memória, o que é do ponto de vista da memória muito ineficiente mas do ponto de vista de processamento muito eficiente (Kwok & Nithianandan, 2006).

O grande problema do GML4J é que é necessário aumentar o tamanho da Heap da JVM ao analisar documentos de dimensões superiores a aproximadamente 10Mb.

2.7.2. XMLBeans

XMLBeans foi inicialmente um projecto Open Source desenvolvido pela BEA para a sua plataforma Weblogic que foi posteriormente liberado para a comunidade Open Source. O objectivo de XMLBeans é receber um esquema de XML e compila-lo para um modelo de objectos Java. Isto pode significar poupanças significantes em memória, pois por exemplo, invés de um byte por dígito, em 4 bytes pode-se guardar um ponto flutuante inteiro. Esta abordagem opõem se ao DOM que guarda os dados num modelo neutro, onde cada elemento/atributo é uma String, o que para dados GML pode ser extremamente eficiente no que diz respeito a memória, considerando que os pontos coordenada são representados por pares de pontos flutuantes em ASCII (Kwok & Nithianandan, 2006).

Como um dos objectivos do XMLBeans é a eficiência, os objectos não são criados até que sejam necessários, o que reduz dramaticamente a memória necessária. Isto implica contudo um tempo maior de acesso visto ter que se efectuar o XQuery/XPath (Kwok & Nithianandan, 2006).

Também, quando comparado com ferramentas como JAXB, XMLBeans suporta de um modo completo os esquemas e conjuntos de informação de XML (Chopra, 2004). Quer isto dizer, que quando um documento XML é reescrito, o documento é totalmente preservado (incluindo comentários).

2.7.3. JAXB

JAXB (Java Architecture for XML Binding) funciona de modo similar a XMLBeans, contudo já vem incluída na versão Java SE 1.6.

A grande diferença entre XMLBeans e outras estratégias Java/XML (tais como JAXB) está no facto de ser possível aceder à toda a informação existente no ficheiro XML. Basicamente em JAXB depois de converter o XML para objectos Java, descarta-se toda a informação que não é usada (e.g. comentários). O XMLBeans apresenta uma abordagem diferente, em vez disso, o XML é guardado, preservando assim toda a informação e dando a possibilidade ao utilizador de trabalhar com as classes geradas ou directamente com XML.

Em suma, XMLBeans e JAXB são abordagens excelentes no que diz respeito a eficiência de memória, mas mais lentos quando se trata de aceder aos dados relativamente ao GML4J. Contudo o GML4J pode ser problemático no que diz respeito à memória em caso de ficheiros grandes.

Na implementação do geobrowser 3D será usado o JAXB, pois tem boa performance e evita o aumento do tamanho da aplicação final, visto que vem incluído com o J2SE.

2.7.4. Optimizações/Simplificações

Um dos mais graves problemas nos dados geoespaciais é por vezes o nível de detalhe que algumas das construções geográficas apresentam. Como os pontos em GML são representados como ASCII, e cada

par de Latitude/Longitude é representado por 35 bytes, a redução de 50% dos pontos reduziria o tamanho do ficheiro GML quase na mesma percentagem.

Há alguns algoritmos que podem executar reduções de pontos que formam geometrias como linhas:

- Algoritmo de Redução de vértices – Os vértices mais próximos que uma tolerância fixa são eliminados forçando assim as extremidades entre cada dois vértices a ter pelo menos a distancia de tolerância.
- Algoritmo de Douglas-Peucker - Em vez de medir as distâncias entre pontos como o algoritmo de redução de vértices, a eliminação de pontos numa linha é determinado pela medida entre pontos na linha e um segmento de linha (Sunday, 2004). Isto é, de certo modo verificasse o ângulo de curvatura entre pontos.
- Combinação de algoritmos - Douglas-Peucker algoritmo que é principalmente usado para refinamento. Tipicamente, o resultado do algoritmo de redução de vértice é usado como input do algoritmo Douglas-Peucker (Kwok & Nithianandan, 2006).
  A cada passo, o algoritmo tenta aproximar uma sequência de pontos por um segmento de recta ligando o primeiro ao último ponto. Uma vez encontrado o ponto mais distante entre a curva e o segmento de recta, verifica-se se sua distância está dentro de um limite estabelecido. Se a resposta for positiva, a aproximação é aceite, caso contrário o algoritmo é aplicado recursivamente às subsequências anteriores e posteriores ao ponto mais distante.

Para que o leitor tenha uma ideia do quanto extensa uma geometria pode ser, um ficheiro que delimita a área de Alto Adige em Itália, pode ser composto por mais de 100 mil pontos. Uma quantidade elevada de pontos para além de aumentar o tamanho do ficheiro GML cria também dificuldades de rendering em run-time.

2.8. Sistemas livres e de código aberto para áreas de Geotecnologias

A evolução tecnológica tem exigido investimentos cada vez maiores na área de Tecnologia da Informação (TI) resultando em muitos casos na compra contínua de sistemas proprietários cada vez mais incómodos devido a restrições e aos custos que impõem. Também o valor elevado e as restrições nas licenças necessárias para a construção de uma infra-estrutura integrada inviabilizam muitos projectos em inúmeras empresas privadas e instituições públicas. Contudo são as instituições públicas que mais sofrem com o modelo de negócios das empresas de soluções proprietárias pois são justamente as principais utilizadoras dos produtos cartográficos (Uchoa & Ferreira, 2004).

A área de Geotecnologias, durante vários anos, esteve dominada por soluções de elevado custo e formatos proprietários. Dois recentes movimentos mudaram este quadro abrindo um novo leque de opções, principalmente para os Sistemas de Informação Geográfica (SIG). Estes movimentos são: a criação do consórcio internacional Open Geospatial (Open Geospatial Consortium) e a revolução do software livre.

Actualmente existe uma quantidade significativa de produtos SIG Open Source para preencher todos os níveis da infra-estrutura de dados espaciais do OGC (Open Geospatial Consortium, 2008). Muitos dos produtos existentes estão inclusive a entrar numa fase de refinamento, podendo oferecer uma alternativa completa a qualquer software proprietário. Contudo muitos outros são pequenos projectos independentes e desenvolvidos de acordo com as necessidades específicas do autor ou utilizador, e

outros não providenciavam muito suporte ou documentação (Kwok & Nithianandan, 2006; Refractions Research, 2004).

Este software Open Source pode ser categorizado em duas grandes tribos de desenvolvimento que podem ser descritas como: A tribo 'C', constituída por projectos como UMN Mapserver, GRASS, GDAL/OGR, OSSIM, Proj4, GEOS, PostGIS e OpenEV e a tribo 'Java', principalmente suportada por projectos como GeoServer, GeoTools, JTS Topology Suite, JUMP/JCS e DeeGree (Refractions Research, 2004).

No que diz respeito a base de dados, existe o projecto PostGIS/PostgreSQL (PostGIS, 2001) que devido ao facto de conter interfaces tanto em C/C++ como em Java é mais ou menos igualmente usado em ambas as tribos. Também existem casos em que alguns projectos da tribo 'C' existem ou são portados para outras linguagens, no caso do Java através do JNI (Java Native Interface).

Os projectos 'C' são em geral, mais maduros que os projectos em Java, pois tem estado em desenvolvimento por um período mais longo de tempo e tem tido mais tempo para atrair as comunidades de desenvolvimento activas, acabando geralmente por no fim apenas se criar portes para outras linguagens (Refractions Research, 2004).

No mundo do "Java" existem várias tentativas independentes de criar ferramentas completas como o OpenMap, GeoTools e DeeGree, contudo alguns deles como por exemplo o GeoTools e DeeGree estão a trabalhar juntos numa convergência. Nesta tribo quase todos os projectos usam JTS Topology Suite para a representação da geometria.

Existem também outros projectos paralelos como GML4J e WKB4J que são usados directamente em muitos projectos e aplicações.

2.8.1. Software gratuito, de código aberto e livre

Para os menos familiares, a primeira ideia que vem à cabeça quando se fala em SL é a de ser gratuito. A ideia não está errada, mas é muito limitada. É importante perceber o significado e implicações destes termos de modo a evitar que se construa um software inviável devido ao facto de por exemplo, algumas partes do código estarem sobre licenças incompatíveis.

Para começar a organizar estes novos conceitos, serão citadas algumas categorias de softwares de acordo com a liberdade de uso. A forma escolhida para transmitir estes conceitos foi a de começar do mais "livre" para o mais "restrito".

2.8.1.1. Software livre

O primeiro conceito que deve ser compreendido é o seguinte: o SL é um programa de computador como qualquer outro programa proprietário. Tem por isso a mesma finalidade, ou seja, é direccionado para responder a determinados requesitos como, por exemplo: editores de textos, editores de imagens, etc. O que o torna diferente é o tipo de licença, que deve garantir ao SL, segundo a Fundação Software Livre (Free Software Foundation), quatro liberdades:

- A liberdade de executar o programa, para qualquer propósito;

- A liberdade de estudar como o programa funciona e adaptá-lo para as suas necessidades. Acesso ao código fonte é um pré-requisito para esta liberdade;
- A liberdade de redistribuir cópias, permitindo a ajuda ao próximo;
- A liberdade de aperfeiçoar o programa, de modo que toda a comunidade beneficie. Acesso ao código fonte é um pré-requisito para esta liberdade.

O conceito de SL está associado a um grande movimento social, onde a ideia de liberdade do uso do software é pregada como solução do problema gerado pela limitação do conhecimento tecnológico imposta pelos sistemas proprietários (Uchoa & Ferreira, 2004).

2.8.1.2. Software de código aberto

Segundo Uchoa (Uchoa & Ferreira, 2004), nesta categoria o utilizador tem acesso:

> "... ao código fonte, podendo altera-lo para atender às suas necessidades. Muitas vezes, as ideias de SL e código aberto confundem-se no nosso quotidiano. A melhor forma de compreender a diferença entre elas é observar que, normalmente, o software de código aberto não respeita alguma(s) das quatro liberdades do SL (ver item anterior). Outra forma de analisar esta diferença é pensar que "o código aberto faz alusão a uma metodologia de desenvolvimento, enquanto o software livre está relacionado a um movimento social ..."

2.8.1.3. Software gratuito

Uchoa (Uchoa & Ferreira, 2004) define o software desta categoria como:

> "... disponibilizado de forma gratuita, mas que normalmente, não pode ser modificado e não disponibiliza o código fonte.
>
> É possível também que a licença impeça a redistribuição do mesmo. Também são conhecidos como Freeware. Cuidado para não confundir com os Shareware, pois estes últimos apesar de também serem gratuitos, possuem alguma limitação funcional em relação ao software original."

Existem ainda outras formas de classificação do software envolvendo definições adicionais como o software semi-livre, o software proprietário e o software comercial (Uchoa & Ferreira, 2004).

2.8.1.4. Software semi-livre

> "É o software que não é livre, mas que permite: a utilização, a cópia, a modificação e a distribuição (incluindo a distribuição de versões modificadas) para fins não lucrativos. PGP é um exemplo de programa semi-livre. Os sistemas desta categoria não podem ser incluídos em sistemas operativos livres, ou seja, não podem acompanhar uma distribuição GNU/Linux".
>
> (Uchoa & Ferreira, 2004)

2.8.1.5. Software proprietário

Estes sistemas normalmente são protegidos por algum tipo de patente. O seu uso, redistribuição e modificação é proibido ou requer uma permissão. Um software proprietário pode ser ou não feito com finalidades comerciais, isto é, com intenção de ser vendido.

2.8.1.6. Software comercial

> "É o software desenvolvido por uma empresa que visa obter alguma forma de lucro. Apesar de softwares comerciais e proprietários estarem muitas vezes associados, pertencem a categorias diferentes. Existem softwares livres que são comerciais, assim como existe software de código aberto que também é comercial ..."
>
> (Uchoa & Ferreira, 2004)

2.8.2. Licenças de código aberto

Como diz Uchoa e Ferreira (Uchoa & Ferreira, 2004):

> "As licenças BSD e Apache são duas das mais antigas licenças de código aberto. São também licenças que ilustram bem alguns princípios básicos das licenças de código aberto. Estas licenças, juntamente com a MIT (também conhecido por X), são licenças de código aberto clássicas para o licenciamento de softwares e são utilizadas em muitos projectos de código aberto. Alguns exemplos bem conhecidos de sistemas baseados nestas licenças são: o servidor Apache HTTP e os sistemas operativos BSDNet e FreeBSD"

De modo a ilustrar a importância das diferenças entre as licenças, deve-se observar o que ocorre quando elas são aplicadas a um determinado código fonte.

O primeiro facto é que este código poderá ser utilizado para compor um sistema proprietário, sendo que não é exigido que versões de código aberto deste sistema sejam distribuídas. Isto significa que os sistemas abertos criados sob estas licenças podem se transformar em sistemas fechados, gerando uma perda para a comunidade de código aberto. Porém, justamente por este facto estas licenças são bastante flexíveis e compatíveis com quase todo o tipo de licença de código aberto.

2.8.2.1. Licença MIT (ou X)

> "É uma licença de código aberto relativamente simples. A licença dá liberdade total (sem restrições) de uso, cópia, modificação, publicação, distribuição e também permite a venda de cópias do programa. Assim como é comum nas licenças abertas, existe, no texto da licença, a informação de que não há qualquer tipo de garantia pelo uso do software ou por qualquer tipo de dano que o mesmo possa causar, deixando o autor livre de tal responsabilidade ..."

(Uchoa & Ferreira, 2004)

Muitas pessoas a esta isenção de garantia como algo negativo nas soluções SL, contudo se verificarem as licenças proprietárias tais como as do Microsoft Windows, é possível verificar as mesmas cláusulas de isenção através das afirmações claras de que não assumem qualquer tipo de prejuízo.

2.8.2.2. Licença BSD

> "Esta licença é um pouco mais restritiva do que a MIT. Existem inúmeras formas similares a ela, como, por exemplo, a UCB/LBL. Alguns detalhes que devem ser destacados nesta licença são as condições de uso dos binários, códigos fontes e modificações. Os direitos de autor e as condições da licença deverão ser incluídos na distribuição do código fonte. No caso da distribuição de binários, deve-se incluir também os termos da licença em toda a documentação e em outros produtos inclusos nesta distribuição. É também imperativo que não se pode utilizar o nome da instituição e/ou o nome dos autores para promover algum sistema derivado sem uma autorização por escrito dos mesmos. Essa proibição é a diferença mais substancial entre esta licença e a licença MIT"
>
> (Uchoa & Ferreira, 2004)

2.8.2.3. Licença Apache

> "Esta licença é bastante similar às duas citadas anteriormente. A versão 1.1 da licença Apache segue as mesmas premissas da BSD em relação a distribuição e modificação, apresentando um texto relativamente sem restrições. A licença Apache v1.1, apesar de possuir um texto um pouco mais longo que as anteriores, segue basicamente as mesmas ideias como: copiar, distribuir, modificar, respeitar a autoria, etc. A licença Apache v2.0 é uma revisão da versão anterior e possui diferenças substanciais com relação aos direitos de patentes e às licenças de sistemas derivados desta versão (2.0). Assim como em outras licenças abertas, na Apache não existe a obrigação do utilizador disponibilizar algum sistema derivado de um software desenvolvido sob esta licença, seja através de uma licença de código aberto ou de software livre."
>
> (Uchoa & Ferreira, 2004)

2.8.3. Licenças de softwares livres

Nos tópicos anteriores, foram abordadas as 4 liberdades do SL e agora é a vez das principais licenças que garantem estas liberdades, isto é, garantem que o código será sempre aberto. Para iniciar a abordagem, tem que se entender a mais importante licença livre: a GNU GPL. Além das 4 liberdades do SL, essa licença possui o que podemos chamar de efeito contaminante. Isto significa que um sistema derivado de algum software sob a licença GPL vai ter que ser licenciado sob a mesma licença. Quando não for

possível "herdar" a licença GPL por algum motivo (royalties, patentes, decisão judicial, etc.), o sistema não poderá utilizar aquele software GPL. Caso o sistema já tenha sido desenvolvido e venha a ter alguma limitação contrária a licença GPL, o mesmo não poderá ser utilizado, comercializado, distribuído, etc. Este efeito contaminante justifica um pouco porque os SL evoluíram tão rapidamente de simples bibliotecas a complexos sistemas corporativos (Uchoa & Ferreira, 2004).

> "Apesar do ideal de liberdade estar bem protegido na licença GPL, em projectos mais complexos ela pode causar dificuldades para os desenvolvedores. Um bom exemplo disso é um sistema que seja derivado de softwares de código aberto e softwares livres. Se apenas uma das bibliotecas for GPL, todo o sistema terá, obrigatoriamente, que ser GPL. Isso pode inviabilizar alguns projectos, pois a licença GPL é incompatível com as licenças abertas. Para contornar esse problema, o projecto GNU lançou a licença GNU LGPL. Sob esta licença, podemos combinar SL (LGPL) com softwares abertos, sem a exigência de que o novo sistema deva ser licenciado sob a LGPL.
>
> Algo que vale a pena destacar em termos de licenciamento, não somente de softwares, mas de outras formas de expressões artísticas (afinal, criar software também é fazer arte), é o trabalho da Creative Commons. Baseado justamente no projecto GNU, a Creative Commons (CC) tem ajudado muitos autores a divulgar os seus trabalhos apresentando opções de licenciamento de uma forma mais legível para os utilizadores que não possuem muita afinidade com as questões legais"
>
> (Uchoa & Ferreira, 2004)

2.8.4. Servidores

Os seguintes itens apresentam vários projectos Open Source de grande relevância relativamente ao tópico desta tese. Em alguns casos, alguns dos projectos serviram de base ou são usados na implementação da mesma.

2.8.4.1. DeeGree

DeeGree (DeeGree, 2009), anteriormente conhecido como "JaGo" foi desenvolvido inicialmente num ambiente académico na Universidade de Bonn na Alemanha em 1997 (Refractions Research, 2004; Kwok & Nithianandan, 2006). A sua arquitectura é um sistema de passagem de mensagens, desenhada para ser extremamente modular (Refractions Research, 2004).

Antes de deixar o mundo académico, o projecto DeeGree já contava um número considerável de funcionalidades como por exemplo as implementações das especificações WMS (Open Geospatial Consortium, 2008), WFS (Open Geospatial Consortium, 2008) e WCS (Open Geospatial Consortium, 2009), assim como suporte para diversas fontes de dados tais como shapefiles, RDBMS e formatos dados OpenGIS (WKB e WKT), entre outras funcionalidades quer completas e parcialmente completas (Refractions Research, 2004).

Um aspecto também importante de referir é que as implementações Web Map Service 1.3 (Open Geospatial Consortium, 2008), Web Map Service 1.1.1 (Open Geospatial Consortium, 2008) e Web

Coverage Service 1.0.0 (Open Geospatial Consortium, 2009) são referenciadas pela iniciativa OGC CITE (Compliance and Interoperability Testing Initiative) (Open Geospatial Consortium, 2009).

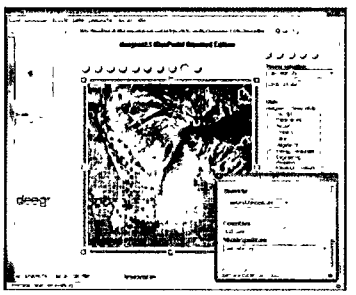

Figura 4 – Interface do DeeGree (DeeGree, 2009)

Responsável Principal: lat/lon e GIS Research Group of the Department of Geography of University of Bonn

Site Principal: http://www.deegree.org

Linguagem (código fonte): Java

Licença: LGPL

2.8.4.2. UMN Mapserver

O "Universidade de Minnesota Mapserver" (geralmente chamado só por "Mapserver") é um servidor de mapas web que faz o render das diversas fontes de dados em mapa cartográficos on-the-fly. Inicialmente desenvolvido pela Universidade de Minnesota, MapServer é actualmente desenvolvido pela comunidade open source.

MapServer pode ser executado na maioria dos sistemas operativos assim como também na maioria dos servidores web. Fornece uma API completa que pode ser acedida através de Python, Perl, PHP, Java e C (linguagem nativa). Tem uma comunidade multi-disciplinar, com equipas que dedicam 100% do seu tempo à manutenção do produto, documentação significativa e um processo de lançamento transparente.

Em méritos técnicos, suporta mais fontes de dados que a maioria dos produtos proprietários, tem melhor desempenho e é simples de instalar e configurar. Suporta também alguns padrões e base de dados espaciais industriais, classificação de características on-the-fly e padrões da Open Geospatial Consortium (OGC) tais como WMS, WFS e WCS. Faz uso de tecnologias open-source como GDAL/OGR, PostGIS e PROJ.4 e integra ambientes front-end como ka-Map, Chameleon, Mapbender, MapBuilder e Cartoweb.

Site Principal: http://mapserver.gis.umn.edu

Linguagem (código fonte): C

Licença: MIT-style

Padrões OGC: SFS (PostGIS), WFS (parcial), WMS e GML

2.8.4.3. GeoServer

O projecto GeoServer (GeoServer, 2008) é um sistema indicado para o desenvolvimento de aplicações de SIG para a WEB. O GeoServer é implementado em J2EE e contempla as principais especificações do OGC (Open Geospatial Consortium, 2008; Uchoa & Ferreira, 2004).

GeoServer é construído sobre biblioteca GeoTools e como resultado, muito da lógica interna do servidor (fontes de dados, analisador gramatical de GML, suporte de filtros XML, etc.) na verdade reside e é mantida ao nível da biblioteca GeoTools (Refractions Research, 2004).

Quando comparado com o MapServer (MapServer, 2008), conclui-se que possui menos recursos, mas possui um ponto forte a favor: as suas implementações do padrão WFS 1.1 e 1.0.0 (Open Geospatial Consortium, 2008) foram escolhidas pelo OGC (Open Geospatial Consortium, 2008) como uma referência, sendo citado no portal CITE (OGC Compliance & Interoperability Testing & Evaluation) (Open Geospatial Consortium, 2009). Como uma implementação de referência, ao GeoServer será exigido suportar todos os aspectos da corrente e próximas especificações (Refractions Research, 2004; Uchoa & Ferreira, 2004).

O GeoServer apresenta suporte para ESRI® Shapefiles (ESRI, 1998), ESRI® ArcSDE (ESRI, 2008), PostgreSQL/PostGIS (PostGIS, 2001) e Oracle® Spatial (Oracle, 2009).

Responsável Principal: The Open Planning Project (TOPP)

Site Principal: http://geoserver.org

Linguagem (código fonte): Java

Licença: GPL 2.0

Padrões OGC: SFS (PostGIS, somente manipulação de feição), WFS, WMS, WPS e GML

2.8.5. Aplicações Desktop

As aplicações que se seguem representam os projectos open source mais relevantes no que diz respeito a aplicações SIG desktop.

2.8.5.1. MapBuilder

MapBuilder (Mapbuilder, 2009) é um cliente que corre num web browser e é compatível com alguns padrões OGC. É capaz de renderizar mapas provenientes de Web Map Services (Open Geospatial Consortium, 2008), Web Feature Services (Open Geospatial Consortium, 2008), GeoRSS e Google Maps. Suporta a edição de mapas através do WFS-T, e usa uma interface interactiva em AJAX.

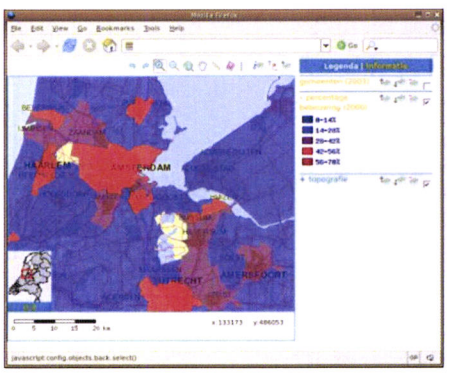

Figura 5 – Interface do MapBuilder (Mapbuilder, 2009)

Site Principal: http://communitymapbuilder.osgeo.org

Linguagem (código fonte): Java

Licença: LGPL

2.8.5.2. MapGuide Open Source

MapGuide Open Source (MapGuide, 2008) é uma plataforma Web-based que permite aos utilizadores desenvolver depressa e publicar aplicações cartográficas ou serviços geoespaciais web.

MapGuide é constituído por um visualizador interactivo que inclui suporte para selecção de características, inspecção de propriedade e operações como buffer, seleccionar dentro de (within) e de medida, entre outras.

MapGuide inclui uma base de dados em XML para gerir o conteúdo e suporta a maioria dos populares formatos de arquivo geoespacial, bases de dados e padrões.

MapGuide oferece ainda uma API para PHP, .NET, Java e JavaScript para o desenvolvimento da aplicação.

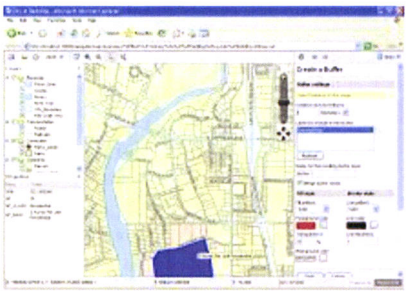

Figura 6 – Interface do MapGuide Open Source (MapGuide, 2008)

Site Principal: http://mapguide.osgeo.org

Linguagem (código fonte): Java

Licença: LGPL

2.8.5.3. OpenMap

OpenMap foi desenvolvido originalmente através da BBN Technologies para projectos de consultadoria com origens que datam a 1987. Foi um dos primeiros projectos Open Source em Java em informações geográficas e por isso, por vezes o seu código base é um pouco confuso. A velha arquitectura foi mantida contudo vários conceitos novos e modos de aceder aos dados foram adicionados ou revistos.

OpenMap ainda é desenvolvido activamente pela BBN que fornece contractos de suporte para companhias que querem usar OpenMap como parte de um produto ou outro desenvolvimento.

Site Principal: http://openmap.bbn.com

Linguagem (código fonte): Java

Licença: Mozilla-style

2.8.5.4. OpenLayers

OpenLayers (OpenLayers, 2008) é uma biblioteca puramente em JavaScript para exibir mapas. OpenLayers implementa uma API JavaScript para construir aplicações geográficas web-based semelhantes às do Google Maps e MSN Virtual Earth apenas com a diferença que é Open Source e é desenvolvido por uma larga comunidade Open Source.

Site Principal: http://openlayers.org

Linguagem (código fonte): Java

Licença: BSD-style

2.8.5.5. GRASS GIS

O GRASS (Geographic Resources Analysis Support System) (GRASS GIS, 2008) é claramente o mais velho de todos os produtos de SIG Open Source (Uchoa & Ferreira, 2004). Foi originalmente um projecto não Open Source desenvolvido pelo US Army Construction Engineering Research Laboratories (USA-CERL), começado em 1982 para providenciar capacidades que não existiam nos sectores comerciais de SIG. O Exército manteve GRASS debaixo de desenvolvimento activo até 1992 e continuou lançar patchs até 1995. GRASS foi apanhado pela comunidade académica em 1997, quando a Universidade de Baylor começou a coordenar o desenvolvimento, passando a ser Open Source oficialmente em 1999 sobre GPL.

GLASS é actualmente usado em projectos académicos e comerciais em todo o mundo, como também em muitas agências governamentais com NASA, NOAA, USDA, DLR, CSIRO, National Park Service, U.S. Census Bureau, USGS, e outras empresas de consultadoria.

Originalmente escrito como um sistema de análise de imagem raster, GRASS passou a ter capacidades de análise de vectores. GRASS pode importar uma gama extensiva de formatos através do uso de bibliotecas com GDAL e OGR. GRASS também tem a habilidade para ler atributos directamente de base de dados espaciais como PostGIS / PostgreSQL.

Actualmente consiste em mais de 350 módulos para processamento vectorial (2D/3D), raster e dados de voxel.

Ao contrário do Thuban e do JUMP no qual todo o trabalho do utilizador está baseado num ambiente gráfico amigável, o utilizador do GRASS precisa recorrer à linha de comandos para ter acesso a alguns recursos.

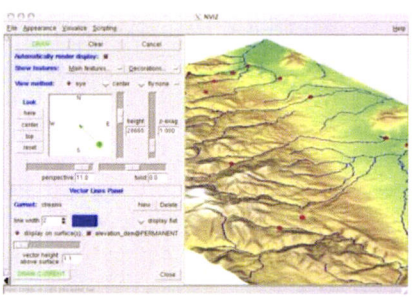

Figura 7 – Interface do GRASS GIS (GRASS GIS, 2008)

Site Principal: http://grass.osgeo.org/

Linguagem (código fonte): C

Licença: GPL

Padrões OGC: SFS (conexão ao PostGIS feita através do PostGRASS)

2.8.5.6. gvSIG

O gvSIG (Conselleria d'Infraestructures i Transport, 2008) é um Sistema de Informação Geográfico (SIG), ou seja, é uma aplicação desenhada para capturar, armazenar, controlar e analisar usando qualquer tipo de informação geográfica de modo a resolver problemas complexos de administração e planeamento. O gvSIG é conhecido por ter uma interface amigável, que permite aceder aos formatos mais comuns, vector & CAD (shapefile, GML, KML, DGN, DXF, DWG), raster (ecw, ENVI hdr, ERDAS img, (Geo)TIFF, GRASS,..), base de dados (PostGIS, MySQL, Oracle, ArcSDE) ou remotos (ECWP, ArcIMS). Tem uma gama extensiva de ferramentas para trabalhar com informação geográfica (ferramentas de navegação, de query, CAD, processamento raster, geoprocessing, redes, etc.), que transforma gvSIG na ferramenta ideal para utilizadores que trabalham na área de terreno. Permite o uso de padrões como WMS, WFS (e WFS-T), WCS, WMC, a descoberta de serviços e catálogos tais como: Z3950, SRW, CSW (ISO/19115 e ebRIM.

É desenvolvido com os princípios de INSPIRE em mente, e tem como objectivo substituir o ArcView.

Site Principal: http://www.gvsig.gva.es

Linguagem (código fonte): C

Licença: GPL 2.0

2.8.5.7. JUMP (Java Unified Mapping Platform)

Este sistema é um Framework Java para o desenvolvimento de aplicações de SIG. Foi desenvolvido por uma empresa canadense e tornou-se muito popular principalmente pelo ambiente gráfico bem amigável, pela excelente documentação e pela facilidade de programar novas funcionalidades. Neste ambiente orientado a objectos, uma característica muito interessante para instituições que estão em fase de migração é a flexibilidade de poder ser executado em qualquer plataforma (The JUMP Unified Mapping Platform, 2003; Uchoa & Ferreira, 2004).

Internamente o Framework é composto por uma biblioteca denominada JTS (veja o item 2.8.6.8 abaixo) que implementa o padrão SFS (OGC), permitindo inúmeras análises topológicas sobre geometrias em 2D (Refractions Research, 2004; Uchoa & Ferreira, 2004).

Com o crescimento da comunidade, novos plugins estão sendo desenvolvidos e disponibilizados livremente na Internet, permitindo expandir as funcionalidades da ferramenta.

Responsável Principal: Vivid Solutions e Refractions Research

Site Principal: http://www.jump-project.org

Linguagem (código fonte): Java

Licença: GPL

Padrões OGC: SFS, WMS, WFS, WPS e GML (muitos através de plugins).

2.8.5.8. OpenEV

OpenEV (OpenEV, 2006) é uma aplicação para visualizar dados geográficos, originalmente criada para um ambiente de Linux mas que recentemente portada para Windows. A funcionalidade mais interessante do OpenEV (OpenEV, 2006), é que foi implementada recorrendo ao OpenGL para visualizar os dados. Isto significa que o OpenEV (OpenEV, 2006) pode apresentar um desempenho de rendering muito bom, mas também restringe as plataformas nas quais pode ser executado. O OpenEV (OpenEV, 2006) pode ver arquivos de imagem muito grandes e criar visões das imagens em 3D quando combinado com arquivos de elevação digitais.

Site Principal: http://openev.sourceforge.net

Linguagem (código fonte): C / Python

Licença: LGPL

2.8.5.9. OSSIM

O Software Image Map (OSSIM, 2008) é uma biblioteca em C++ assim um motor de alta performance para sensoriamento remoto, processamento imagem, fotometria, com várias aplicações construídas por cima. O seu desenvolvimento tem estado activo desde 1996, tendo sido fundado por várias agências governamentais dos Estados Unidos da América na comunidade de inteligência e defesa. O benefício técnico primário de OSSIM é que é uma arquitectura que permite dividir tarefas de processamento de imagens em componentes independentes e que podem ser executadas em paralelo. Como resultado essas tarefas podem ser executadas em arrays de computação de alto desempenho, como clusters Beowulf aumentando de um modo incrível a performance.

O OSSIM é desenvolvido principalmente por ImageLinks e é usado interiormente pela companhia para muitas tarefas de produção e manipulação de imagens raster em cadeia. ImageLinks também usa OSSIM na sua linha de produto RasterWare.

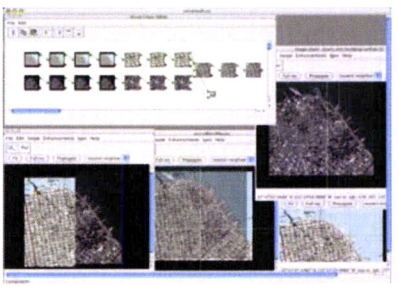

Figura 8 – Interface do OSSIM (OSSIM, 2008)

Site Principal: http://www.ossim.org

Linguagem (código fonte): C++

Licença: GPL

2.8.5.10. Quantum GIS

O Quantum GIS foi criado em 2002, quando Gary Sherman começou a procurar por um visualizador de dados geográficos para Linux. Devido a isso é construído principalmente para correr em desktops com Linux. QGIS depende do popular QT (Q de QGIS), também usado por exemplo pelo KDE. Porém, QT está disponível para outras plataformas (Win32, OS / X, Solaris).

Apesar de QGIS na sua primeira versão só suportar camadas de PostGIS, agora suporta também fontes de dados provenientes de Shapefiles. QGIS usa OGR como uma ponte para importação de dados, suportando assim todos os formatos de OGR. Suporta também uma integração com GRASS GIS, ferramentas de digitalização, OGC WMS e WFS, bookmarks espaciais, visualização, edição de atributos, etc.

Com uma comunidade em constante crescimento, este projecto também contempla o padrão SFS (OGC).

Responsável Principal: Gary Sherman

Site Principal: http://qgis.org

Linguagem (código fonte): C++

Licença: GPL

Padrões OGC: SFS (PostGIS)

2.8.5.11. TerraView

Este sistema é um visualizador de bases cartográficas voltado para aplicações de SIG, que possui uma interface amigável e capacidade de manipular dados vectoriais (pontos, linhas e polígonos) e matriciais (grades e imagens). Foi desenvolvido pelo Instituto Nacional de Pesquisas Espaciais (INPE) no Brasil através do uso da biblioteca TerraLib (veja o item 2.8.6.10 abaixo) (Uchoa & Ferreira, 2004).

Apesar de trabalhar com o PostgreSQL, o TerraView não segue a especificação SFS (OGC), trabalhando com uma estrutura de dados própria.

Responsável Principal: INPE

Site Principal: http://www.dpi.inpe.br/terraview/index.html

Linguagem (código fonte): C++

Licença: GPL

Padrões OGC: Nenhum

2.8.5.12. Thuban

Thuban é um visualizador extensível e multi-plataforma para dados de SIG escrito em Python (Uchoa & Ferreira, 2004), usando o WxWindows para a interface gráfica. O sistema possui uma interface amigável e alguns recursos úteis, tais como:

- Suporte a dados vectoriais: ESRI® Shapefiles e conexão PostGIS;
- Suporte a dados matriciais: GeoTIFF;
- Permite análises (queries) e junções (joins) de tabelas;
- Possui suporte a projecções;
- Ferramenta de impressão e para exportação de vectores;

Assim como o JUMP, este sistema é facilmente expansível através de plugins.

Responsável Principal: Intevation GmbH (thuban@intevation.de)

Site Principal: http://thuban.intevation.org

Linguagem (código fonte): Python

Licença: GPL

Padrões OGC: SFS (PostGIS)

2.8.5.13. uDig / JUMP2

O uDig (uDig, 2008) é um projecto que aproveita os potenciais do projecto GeoTools (GeoTools, 2008) (design, estruturas de dados, padrões) e os potenciais do projecto JUMP (UI, renderer, interactividade) em um editor de desktop novo capaz de interagir com uma gama de fontes de dados na rede ou internet (Kwok & Nithianandan, Visualization of GML Data in OpenMap, 2006; Refractions Research, 2004).

uDig significa "User-friendly Desktop Internet GIS" e tem como objectivo trazer tecnologias de cartografia web como WMS e WFS de um modo transparente para o computador de utilizadores normais, sendo um valioso sucessor do cliente JUMP (Refractions Research, 2004).

Assim sendo, uDig tem as capacidades de um cliente WFS com suporte leitura/escrita, permitindo a edição directa de dados expostos por WFS-T. Suporte WMS e Styled Layer Descriptor (SLD) permitindo visualizar e alterar em dados publicados por WMS.

Permite aos utilizadores abrir, sobrepor, e editar ficheiros Shape e GeoTIFF assim como dados guardados em PostGIS, OracleSpatial, ArcSDE, e MySQL.

Site Principal: http://udig.refractions.net

Linguagem (código fonte): Java

Licença: LGPL

2.8.6. Bibliotecas para desenvolvimento

Estas bibliotecas são exploradas pelas várias aplicações, permitindo assim que as aplicações se foquem no desenvolvimento de funcionalidades mais importantes.

2.8.6.1. FDO

A Tecnologia FDO ("Feature Data Object") Data Access é uma API para manipular, definir e analisar informação geoespacial independentemente de onde é armazenada. FDO usa um modelo fornecedor-baseado para suportar uma variedade de fontes de dados geoespaciais.

A primeira versão de FDO foi lançada com o produto Autodesk Map 3D 2005 pela primavera de 2004. Incluía suporte para Oracle e SDF. A versão 2.0 estendeu o suporte para ArcSDE e a versão 3.0 para MySQL, SQL Server, ODBC, SHP, Raster, OGC WFS e OGC WMS.

A liberação de FDO como fonte aberta coincidiu com a liberação de MapGuide como fonte aberta em 2006. Incluía o SDF, SHP, MySQL, ArcSDE, ODBC, OGC WFS e OGC WMS.

Site Principal: http://fdo.osgeo.org

Linguagem (código fonte): C

Licença: LGPL

2.8.6.2. GDAL/OGR

A Geospatial Data Abstraction Library (GDAL/OGR) é a biblioteca de código aberto mais poderosa no campo de visualização/conversão de formatos matriciais e vectoriais. Ela é amplamente utilizada não somente nos projectos livres, mas também nos sistemas proprietários. Oferece um motor primário para aceder a dados para muitas aplicações como MapServer, GRASS, QGIS e OpenEV. Também é utilizada por pacotes como OSSIM, Cadcorp SIS, FME, Google Earth, VTP, Thuban, ILWIS, MapGuide e ArcGIS. O suporte profissional também está disponível para software comercial que deseja ajuda para a integrar e estender. É composta por dois pedaços de código logicamente separados: GDAL fornece uma biblioteca abstracta para dados raster e módulos para ler e escrever vários formatos raster; OGR fornece uma biblioteca abstracta para dados vectoriais e módulos para ler e escrever formatos vectoriais. Porém, as duas bibliotecas são mantidas dentro do mesmo sistema de construção por razões históricas e porque ambas as bibliotecas são mantidas pela mesma pessoa. Também vem com uma variedade de utilidades de linha de comandos úteis para tradução e processamento de dados.

Porque a licença de GDAL / ORG é BSD, a biblioteca é também usada em vários pacotes SIG proprietários.

GDAL suporta mais de 50 formatos raster e OGR mais de 20 formatos de vector. Uma lista dos formatos suportados por GDAL/OGR pode ser consultada online em http://www.gdal.org/formats_list.html

Responsável Principal: Frank Warmerdam

Site Principal: http://www.remotesensing.org/gdal/

Site Secundário (mirror): http://gdal.maptools.org

Linguagem (código fonte): C++

Outras linguagens suportadas: Perl, Python, VB6 (sem o uso de SWIG), GDAL interface para R, Ruby, Java e C# / .Net

Licença: X/MIT Style

Padrões OGC: GML

2.8.6.3. GEOS

A GEOS (Geometry Engine - Open Source) (GEOS, 2008) é apenas um porte da JTS Topology Suite de Java para C++. Este projecto surgiu de modo a suportar requesitos existentes no código do PostGIS, pois este não contemplava a especificação SFS (Open Geospatial Consortium, 2009) a 100%. A "criação" da GEOS tornou assim possível uma total compatibilidade do PostGIS com a SFS.

Responsável Principal: Refractions Research

Site Principal: http://geos.refractions.net

Linguagem (código fonte): C++

Licença: GPL

Padrões OGC: SFS

2.8.6.4. GeoTools

O Geotools é um projecto em Java que implementa muitas soluções OGC. Tem uma arquitectura modular que permite adicionar e remover facilmente novas funcionalidades. O Geotools tem como objectivo suportar os protocolos do OGC assim como outros padrões relevantes que vão surgindo.

O GeoTools é comparável ao OpenMap. Contudo o OpenMap é desenvolvido independentemente, ao contrário de GeoTools que é desenvolvido à volta de varias bibliotecas GIS como JTS Topology Suite e uDig.

Não é da intenção do projecto GeoTools desenvolver produtos acabados ou aplicações, mas é intenção interagir e suportar completamente outras iniciativas e projectos que gostariam de usar o GeoTools para criar tais recursos.

O Geotools apresenta módulos individuais para as diversas especificações do OGC (por exemplo. Filter, SLD, GML2) e que também suporta a interacção com uma gama extensiva de fontes de dados (por exemplo Shapefile, MIF / MID, PostGIS e MySQL). Cada módulo tem os seus próprios maintainers, que controlam o conteúdo e direcção do módulo.

A manutenção global e direcções futuras de GeoTools são administradas pelo GeoTools Project Management Committee, que inclui um pequeno número de programadores activos que assumem as responsabilidades em conjunto para decisões de desenho e implementação.

Responsável Principal: Comunidade do próprio projecto.

Site Principal: http://www.geotools.org

Linguagem (código fonte): Java

Licença: LGPL

Padrões OGC: SFS e alguns documentos: 01-009, 01-004 e 02-070.

2.8.6.5. GML4J

A GML4J é uma biblioteca para processamento de GML escrita por Galdos Systems como um teste para a tecnologia GML. É ainda usado por várias aplicações Java para processar GML, mas foi largamente substituído a favor as novas tecnologias de parsing por motivos de desempenho. GML4J usa o sistema de análise gramática DOM (Document Object Model) que requer que todos os dados sejam mantidos em memória. Isto pode resultar em grandes usos de memória em caso de conjuntos de dados muito grandes.

Site Principal: http://gml4j.sourceforge.net

Linguagem (código fonte): Java

Licença: Apache

2.8.6.6. GMT

A "Generic Mapping Tools - Ferramentas de Cartografia Genéricas" (GMT) é um projecto desenvolvido num ambiente académico na Universidade de Havai desde 1988.

O GMT é desenvolvido como um conjunto de pequenos programas manipuladores de dados e de geração de gráficos, que podem ser sequenciados e programados em conjunto de modo a criar cadeias de processamento de dados complexos. Por exemplo, aplicações de GMT podem receber dados de sensores, criar uma grade interpolada, contornar a grade e criar arquivos para imprimir.

Site Principal: http://gmt.soest.hawaii.edu

Linguagem (código fonte): C

Licença: GPL

2.8.6.7. WKB4J

WKB4J é uma biblioteca de interpretação do formato WKB (Well-Known Binary) desenvolvida para fornecer uma interligação de alta velocidade entre o Java e fontes de dados espaciais WKB-enable (normalmente RDBMS), isto é, de transformação de dados entre objectos java e outras fontes de dados (usualmente base de dados como o PostGIS). O formato Well-Known Binary (WKB) e Well-Known Text (WKT) são definidos pelo OGC numa tentativa de interoperabilidade entre as aplicações SIG. O WKB4J fornece uma interface "Factory" para facilitar a criação dos vários objectos primitivos geográficos - geometrias de JTS, geometrias PostGIS e geometrias de OpenMap.

Site Principal: http://wkb4j.sourceforge.net

Linguagem (código fonte): Java

Source License: GPL

2.8.6.8. JTS Topology Suite

JTS Topology Suite é possivelmente a biblioteca para análises espaciais sobre geometrias em 2D mais conhecida e usada.

O elemento que faz de JTS uma implementação especial é a implementação de "predicados espaciais". Predicados espaciais são funções que comparam dois objectos espaciais e devolvem um resultado boolean true/false que indica a existência (ou ausência) de uma relação espacial particular. Alguns exemplos de predicados espaciais são Contains, Intersects, Touches e Crosses. A implementação JTS dos predicados é especial no sentido em que as funções são "robustas". Quer dizer, não existe nenhum caso especial de geometrias estranhas ou coordenadas estranhas que são capazes de produzir uma falha ou resultado incorrecto. Esta é uma propriedade sem igual, pois a maioria dos produtos proprietário não inclui predicados espaciais tão robustos.

A JTS também inclui implementações dos "operadores" espaciais que de duas geometrias devolvem um resultado geométrico novo. Exemplos dos operadores incluem Difference, Union e Buffer. Os operadores JTS implementados têm sido amplamente testados, mas não têm as garantias de robustez como os predicados.

As implementações de predicados e operadores espaciais são valiosas porque são extremamente difíceis de programar. É por isso que a biblioteca de JTS é usada amplamente através de outros projectos de OSS. Ao usar a JTS, pode-se adquirir um conjunto de geometrias, com os métodos espaciais mais difíceis já implementados.

Responsável Principal: Vivid Solutions

Site Principal: http://www.vividsolutions.com/jts

Linguagem (código fonte): Java

Licença: LGPL

Padrões OGC: SFS

2.8.6.9. Proj4

É a biblioteca mais utilizada nos sistemas livres (e de código aberto) para tratamento de projecções. A Proj4 foi inicialmente desenvolvida pela USGS e, é actualmente utilizada pelo GRASS GIS, MapServer, PostGIS, OGDI e OGRCoordinateTransformation, entre outros. É uma biblioteca para reprojecção de coordenadas, capaz de executar transformações entre sistemas de projecção cartográficos e também diferentes esferóides e datums. Inclui uma API e aplicações de linha de comandos como *proj*, *cs2cs*, *nad2nad*, *nad2bin* e *geod*. O aplicativo *nad2nad* é um filtro para conversão de dados entre o North

American Datum 1927 (NAD27) e North American Datum 1983, o *nad2bin* é um aplicativo que converte um arquivo NAD no formato ASCII em um arquivo binário utilizado pelo *nad2nad*.

Responsável Principal: Frank Warmerdam

Site Principal: http://trac.osgeo.org/proj/

Linguagem (código fonte): C

Licença: MIT-style

Padrões OGC: Não aplicável

2.8.6.10. TerraLib

É uma excelente biblioteca para o desenvolvimento de aplicações em SIG. Apesar de bastante flexível, esta biblioteca adopta um modelo geográfico de dados diferente do SFS (OGC). Com isso, aplicações baseadas nela "herdarão" esta característica. Este é o caso do TerraView e do TerraCrime, cujas bases não podem ser acedidas pelos vários sistemas livres que seguem o OGC (JUMP, GRASS, Thuban, etc.) (Uchoa & Ferreira, 2004).

Responsável Principal: INPE

Site Principal: http://terralib.dpi.inpe.br/portugues.html

Linguagem (código fonte): C++

Licença: LGPL

Padrões OGC: Nenhum

2.8.7. Catálogos de Metadados

Os catálogos de metadados são serviços que armazenam informação descritiva sobre serviços ou outro tipo de itens. Por essa razão são usados como motores de pesquisa.

2.8.7.1. GeoNetwork

O GeoNetwork é uma aplicação opensource de catálogo para gerir recursos espacialmente referenciados. Fornece um editor de metadados e funções de procura assim como também um visualizador incorporado de mapas interactivo.

O GeoNetwork tem sido desenvolvido por United Nations Food and Agriculture Organization (UN FAO) para ligar as comunidades de informação espacial e os dados que usam, usando uma arquitectura moderna que é ao mesmo tempo poderosa e de baixo custo baseada nos princípios de Open Source Software (FOSS) e International and Open Standards para serviços e protocolos (como ISO/TC211 e OGC).

Site Principal: http://geonetwork-opensource.org

Linguagem (código fonte): Java

Licença: GPL v3

2.8.8. Base de dados

2.8.8.1. PostGIS

O PostgreSQL (PostgreSQL, 1996) foi o primeiro SGBD (Sistema de Gestão de Base de Dados) de código aberto a trabalhar com um módulo específico para o tratamento dos dados geográficos vectoriais.

Este módulo denominado de PostGIS (PostGIS, 2001) foi desenvolvido por uma empresa canadense chamada Refractions Research e segue a especificação SFS (Simple Features Specification) do OGC (Uchoa & Ferreira, 2004).

Para que o PostGIS contemple toda a SFS, é necessário que ele seja compilado juntamente com a biblioteca GEOS (Geometry Engine Open Source). Com isso, o PostGIS passa a possuir mais de 130 funções e operadores para o tratamento de dados geográficos vectoriais. Deste modo, suporta agora funcionalidades que eram antigamente possíveis apenas com o Oracle® Spatial ou Microsoft® SQL Server (com o ArcSDE) (Uchoa & Ferreira, 2004).

O PostgreSQL suporta três tipos de indexação nativos: BTree, RTree e GiST (Generalized Search Trees). O BTree é usado para ordenação de dados em um eixo somente, logo ele não tem muita utilidade para tratamento de dados geográficos. Já o RTree divide os dados em rectângulos que, por sua vez, podem ser novamente divididos em novos rectângulos, e assim sucessivamente. Apesar de o RTree ser utilizado por alguns bancos de dados espaciais para indexação de dados em SIG, a implementação do RTree do PostgreSQL não é tão robusta quanto a implementação GiST. Esta última pode ser entendida de maneira simples como uma divisão dos dados em "objectos ao lado de", "objectos que se sobrepõem a", "objectos que estão dentro de", etc. Assim como as outras indexações, ela é utilizada para acelerar pesquisas, porém ela pode tratar uma variedade de estruturas de dados irregulares, o que não é possível com o BTree. Devido às limitações do RTree do PostgreSQL, o PostGIS emprega a RTree construída sobre o GiST (Uchoa & Ferreira, 2004)

O ponto forte de PostGIS é que se tornou o standard em base de dados espaciais (backend) para muitas ferramentas SIG open source. Como resultado, uma camada em PostGIS pode ser analisada com GRASS, publicada na web com Mapserver, visualizada num desktop com OpenEV e exportada para formatos proprietários com OGR.

Responsável Principal: Refractions Research Inc

Site Principal: http://postgis.refractions.net

Linguagem (código fonte): C

Licença: GPL v2

Capítulo III: Metodologia

3.1. Objectivos do capítulo III

Neste capítulo é apresentada e implementada uma arquitectura orientada para o uso de dados georreferenciados. Pretende-se não só que esta possibilite a visualização de dados georreferenciados num ambiente 3D, mas que esta permita também, de modo interactivo e intuitivo, manipular e analisá-los visualmente de modo a que possam ser ainda mais úteis em processos de suporte à decisão.

O tratamento de dados apresenta-se estruturado em três sub-tópicos:

- Visualização de dados georreferenciados de um modo natural e interoperável
- Manipulação (modificar, apagar e inserir) de dados de modo interoperável
- Processamento dos dados de um modo visual e interoperável

São também sugeridas e implementadas algumas alterações as especificações OGC. Cita-se a adição de duas operações à especificação WPS (StoreNewAlgorithm e GetAlgorithm) e diversas alterações importantes para que a especificação WMS seja mais flexível e eficiente (Novo tipo de organização das camadas por pastas, um modo mais eficaz de declarar o CRS, extensão dos valores suportados para o parâmetro BBOX entre outros).

3.2. Ferramentas

Algumas das ferramentas mais importantes, e que serão usadas na implementação das componentes da arquitectura são:

- **JTS Topology Suite** Usada para estruturar e manipular geometrias.
- **World Wind** Fornece o suporte básico para a navegação e visualização de imagens.
- **Geotools** Oferece suporte para leitura de formatos como Shapefile, rendering de características, reprojecção de coordenadas, leitura e geração de GML e representação de características. Foi também o ponto de partida para a implementação de WFS-T.
- **GeoServer** Servidor que suporta o WFS-T.
- **JBoss Server** Servidor que entre muitas coisas suporta Enterprise Java Beans e clustering.
- **N52º North WPS** Ponto de partida para a implementação do protocolo WPS.
- **JOGL** Porte do OpenGL para Java.
- **Java** Linguagem de programação usada em toda a arquitectura.
- **Apache Commons** Conjunto de bibliotecas que fornecem um suporte simplificado para efectuar operações no disco, na internet, de logging, entre outras.
- **JAI** Para manipulação de imagens.
- **Vecmath** Para operações matemáticas.

3.3. Sistema desenvolvido

A quantidade de dados espaciais disponível nos dias de hoje é tão grande, que não faz muito sentido disponibiliza-los fisicamente com um cliente, até porque iria exigir o uso de hardware mais caro tais como discos de elevada capacidade e processadores mais rápidos para o processamento da informação, implicando indirectamente custos mais elevados para a arquitectura.

Numa analogia ao Google Maps, Microsoft Visual Earth e outros, os servidores podem ser os responsáveis pela actualização, armazenamento e processamento de informação. Deste modo apenas cabe ao cliente pedir a informação aos diversos servidores, se estes a disponibilizam. Como consequência, esta abordagem garante também a consistência dos dados que são partilhados e usados pelos diversos utilizadores, visto que podem ser considerados como centralizados.

De modo a resolver algumas destas questões, e também com o objectivo de suportar operadores que necessitam de ajuda no apoio à decisão em questões relacionadas com dados georreferenciados, é apresentada uma Arquitectura Orientada a Serviços (SOA). Incluído nesta SOA, um geobrowser 3D permite aos utilizadores conectarem-se a diversos repositórios de dados geoespaciais de uma forma interoperável através do uso de alguns dos padrões web do OGC (OWS). Esta funcionalidade é essencial para operadores que tipicamente têm que usar um conjunto de fontes heterogéneas, fontes de informação multidimensionais e que variam também com o tempo. A aplicação permite também a descoberta de serviços que podem ser posteriormente acedidos, editados e usados para processamento de forma interactiva, e num ambiente 3D.

No sistema desenvolvido são também implementadas as modificações mencionadas mais à frente no item 3.4, que incluem a possibilidade de se poder obter mapas delimitados por uma geometria, ou de adiçionar novos algoritmos em run-time, algo que por exemplo, num ambiente de catástrofe, fortemente caracterizado por um ambiente colaborativo entre diversas competências, ajudaria a desenvolver em tempo muito reduzido uma simulação que incorpora contributos das diversas partes. Também foram algumas alterações que permitem optimizar a lista de CRS's, e organizar as diversas camadas.

Sem um sistema deste tipo a maioria das aplicações falhará ao tentar relacionar os diversos campos de conhecimento, pois entre muitas razões não apresentam uma interface simples e universal. Esta abordagem possibilita também que computadores normais tais como laptops possam efectuar todo e qualquer trabalho necessário para realizar uma simulação. Deixando assim de parte a necessidade de ter que transportar um super-computador ou até mesmo de possuir um. Isto acontece porque o computador apenas tem de ser capaz de visualizar dos dados, deixando tudo o que necessita de ser processado para serviços de computação disponíveis online.

3.4. Alterações às especificações do OGC

Nesta secção são apresentadas algumas alterações as especificações do OGC, de modo a que estas se tornem mais flexíveis e optimizadas. Algumas delas permitem inclusive a possibilidade de efectuar melhorias e corrigir falhas de um modo colaborativo.

3.4.1. Elemento CRS

O elemento CRS (ver Anexo II, item 3.10) apresenta algumas falhas que podem ser observadas em diversos serviços públicos disponíveis online. Isto acontece porque o agrupamento das camadas está directamente relacionado com a classificação, levando a que por vezes seja impossível organizar as listas de CRS numa estrutura óptima. Esta falta de optimização pode levar a aumentos no tamanho da resposta do servidor superiores a 500%. Por exemplo, quando na camada raiz não é possível listar 1000 CRS comuns a 1000 sub-camadas, porque uma das camadas suporta apenas 999, ou se reorganiza as camadas, o que pode não ser adequado ou possível (pois altera a classificação), ou se declara os 1000 CRS dentro de cada uma das 999 sub-camadas.

Idealmente deveria ser possível criar grupos de CRSs (CRS_GROUP) que fossem identificados por um ID, que poderia depois ser usado como referência em cada uma das camadas.

Exemplo da declaração de um grupo, que inclusivamente pode estender outros grupos:

```
<CRS_GROUP id = "AX02">
    <CRS> CRS:84 </CRS>
    <CRS> EPSG:26718 </CRS>
    <CRS_GROUP ref="AX01"/>
</CRS_GROUP >
```

E depois re-usado nas sub-camadas através de <CRS_GROUP ref="AX02"/>.

Este problema é grave, pois por vezes a declaração de CRSs representa mais de 70% do ficheiro a transmitir.

3.4.2. Elemento Directory

As camadas podem ser organizadas através do elemento Layer, contudo este tipo de estruturação não possiblilita algumas optimizações, nem oferece um mecanismo especifico para catalogar e classificar camadas. A especificação deveria implementar um elemento análogo ao usado pelo KML, como por exemplo:

```
<Directory>
      <Name lang="EN-en">Rivers</Name>
      <Name lang="PT-pt">Rios</Name>
      <Layer>...
</Directory>
```

Deste modo a alteração da ordem das camadas (por exemplo, por motivos de hereditariedade) não implica uma alteração na classificação. Este mecanismo possibilita também o suporte para diversas línguas.

3.4.3. Bounding Box

Actualmente é impossível efectuar um pedido em que o resultado é apenas limitado, por exemplo, a uma circunferência, contudo é uma funcionalidade que os servidores podem querer implementar, e que por motivos de visibilidade é também relevante para o cliente que deste modo não necessita de pré-processar a imagem.

Por exemplo, o caso de uma circunferência pode parecer ilógico pois a forma natural de uma imagem corresponde a uma bounding box, contudo dependendo da informação a ser apresentada pode ser útil a sua limitação a uma circunferência ou polígono. Assim o resto da imagem pode ser devolvido em transparente de modo a não criar uma sobrecarga de informação.

Como extensão, o elemento BoundingBox deveria de incluir um atributo adicional (supportWKT="1|0") a informar se o servidor suporta a extensão. Assim sendo a declaração do Bounding Box nos metadados do serviço passaria a ser algo como:

<BoundingBox CRS="CRS:84" minx="-180" miny="-90" maxx="180" maxy="90" supportWKT="1"/>

No que diz respeito ao pedido, este poderia como extensão, no caso em que o servidor suportar WKT Well Known Text) ser efectuado através da seguinte sintaxe BBOX=[Geometria em WKT].

Rectângulo: BBOX=-180,-90,180,90 ou BBOX=POLYGON(-180 -90, -180 90, 180 90, -180 -90) [1]
Triangulo: BBOX=POLYGON(-180 -90,-140 -80,-100 -80) [1]

É de notar que WKT também suporta a representação de casos mais complexos como conjuntos de polígonos que inclusivamente podem conter concavidades.

Esta funcionalidade está implementada nesta arquitectura através do uso de mascaras, onde a geometria que delimita o pedido é usada como mascara sobre o resultado – isto é, apenas é preenchida a área correspondente ao interior da geometria delimitadora. Para renderizar a imagem a usar com a mascara é usada a bounding box da geometria delimitadora. Note também, que o uso de filtros GML não possibilita a obtenção do mesmo tipo de resultados que esta funcionalidade, visto que os filtros são usados para filtrar características, o que quer dizer, que por exemplo se alguma parte de uma estrada está incluída no filtro, então todo esse troço de estrada aparecerá na imagem. Esta funcionalidade não considera a topologia de uma característica, mais sim os pixéis a serem incluídos na imagem.

3.4.4. Extensão ao protocolo WPS

Como extensão ao protocolo WPS são sugeridas duas alterações. A adição da operação StoreNewAlgorithm que permite adicionar novos algoritmos à lista de algoritmos disponíveis no servidor como serviços web. Para esta operação é necessário especificar um documento que descreve o

[1] Como o parâmetro contém símbolos restritos deve ser codificado. No caso do primeiro exemplo resultaria POLYGON(-180+-90%2c+-180+90%2c+180+90%2c+-180+-90)

processo segundo o padrão (elemento ProcessDescription) e do seu código fonte. Eventualmente pode estar associado a uma duração, ao fim da qual o algoritmo é removido da lista de processos online.

Analogamente às outras operações do protocolo, as novas operações podem ser acedidas através do protocolo HTTP ou HTTPS no qual se podem impor as questões de segurança, sobre as quais é possível encontrar uma extensa literatura.

De modo a adicionar um novo algoritmo é necessário invocar a operação através de um URL como http://www.someserver.com/caminho/wps efectuando POST dos parâmetros necessários.

Um parâmetro importante e que especifica como é tratada a atribuição do identificador ou a resposta do servidor perante um atributo duplicado é especificado pelo IDGEN, que pode tomar os valores da tabela seguinte. É de notar que o identificador de um processo deverá ser único, pois é usado para identificar unicamente um processo.

Valor para idgen	Acção
GenerateNew (por defeito)	O WPS gerará um identificador único para este algoritmo.
UseExisting	Na resposta a uma operação <StoreNewAlgorithm>, um WPS usará o valor do atributo IDENTIFIER como identificador para o algoritmo. Se o identificador estiver duplicado relativamente a um já guardado, o WPS responderá com uma excepção.
ReplaceWhenDuplicated	Um cliente WPS pode pedir ao serviço para gerar um identificador, de modo a substituir o identificador caso este já exista em vez de apresentar uma excepção.

Tabela 1 - Valores para o atributo idegen da Operação StoreNewAlgorithm

O código fonte é identificado pelo elemento <Source> que por sua vez é composto por uma lista de um ou mais sub-elementos <File>. O elemento <File> especifica a linguagem é que foi programado o código fonte, o conteúdo, o nome, e se é a classe principal.

O exemplo que se segue ilustra o exemplo de um pedido StoreNewAlgorithm.

```
<?xml version="1.0"?>
<wps:StoreNewAlgorithm
    version="1.2.0"
    service="WPS"
    idgen="ReplaceWhenDuplicated"
    expire="Thursday, September 7, 2009 2:22 PM"
    xmlns="http://www.servidor.com/myns"
    xmlns:ows="http://www.opengeospatial.net/ows"
    xmlns:wps="http://www.opengis.net/wps"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www. servidor.com/myns
```

```
                    http://www.opengis.net/wps ../wps/1.2.0/WPS.xsd">

    <ProcessDescription processVersion="2" storeSupported="true" statusSupported="false" sourceAvaliable="true">
      ....
    </ProcessDescription>
    <Source>
        <File name="TestAlgorithm.Java" language="Java" main="true" «>R0lGODlhZABq ... base-64 ...</File>
        <File name="sample.bin">R0lGODlhZABq ... base-64 ...</File>
    <Source>

</wps:StoreNewAlgorithm>
```

A segunda alteração seria dar acesso não só à execução de um processo, mas sim também a possibilidade de poder visualizar o seu código, de modo a permitir que seja possível estender ou melhorar.

Contudo esta funcionalidade não é desejada para todos os processos por diversas razões, sendo por isso apenas possível obter o código fonte de algoritmos cujo seu ProcessDescription especifique o atributo sourceAvaliable igual a "true".

Como resposta do servidor, pode-se obter uma mensagem de erro caso não seja possível obter o código fonte, ou o código fonte. O formato em que se pode receber o código fonte pode também ser especificado pelo servidor através do parâmetro FORMAT.

Um exemplo de um pedido seria:

```
http://www.servidor.com/wps?SERVICE=WPS&VERSION=1.2.0&REQUEST=GetAlgorithm&IDENTIFIER=TestProcess&FORMAT=XML
```

Também estas operações podem implicar perigos para o servidor devido ao facto de poder ser adicionado algum algoritmo malicioso.

Por essas razões é também muito importante assegurar a segurança deste, como medida auxiliar a dos mecanismos de autenticação.

Alguns dos mecanismos que podem ser usados e que são bastante eficazes são por exemplo o uso do SecurityManager.

Através da criação de uma classe que estenda o SecurityManager é possível criar um mecanismo que monitoriza todas as permissões dos processos em execução, que no mais drástico dos casos pode negar o acesso a qualquer recurso, seja este um ficheiro local ou remoto ou acesso a outros objectos/código, ficando este limitado exclusivamente a execução de um algoritmo de processamento.

Um exemplo do uso mais trivial deste mecanismo é:

```
try {
        SecurityManager sm = new SecurityManager();
        System.setSecurityManager(sm);
}
catch (SecurityException se) {
        // SecurityManager already set
}
```

3.5. Arquitectura Orientada a Serviços (SOA)

A Arquitectura Orientada a Serviços (SOA) sugerida como solução a este problema consiste em três componentes lógicas principais: Consumidores, Infra-estrutura SOA e Produtores.

A componente lógica "Consumidores" é composta por dispositivos que fazem uso dos serviços (e.g. desktops, computadores portáteis, dispositivos moveis ou até mesmo outros serviços web que usam os serviços) e são capazes de apresentar uma interface entre os utilizadores e o sistema. Esta componente comunica com a infra-estrutura SOA através de protocolos como o HTTP ou IIOP.

Em contraste, todas as entidades que oferecem serviços, dados ou funcionalidades (motores de busca, ficheiros, Geospatial One-Stop, etc.) que são usadas nos serviços disponibilizados por esta arquitectura são classificados como "Produtores".

Quanto à infra-estrutura SOA, esta pode ser dividida em três sub-camadas:

- Suporte – Providencia suporte base as funções da SOA. Por exemplo o serviço de catalogo CS-W, que permite a descoberta de serviços ou a arquitectura ESB, que pode servir de ponte para as comunicações entre os consumidores e o produtores.
- Serviços – Entidades que efectuam uma tarefa especifica, quando requisitada. Alguns exemplos são os serviços web WMS, WFS.
- Aplicações – Camada composta por aplicações, que tem como objectivo primário a exploração desta arquitectura. Nesta camada é possível encontrar um geobrowser 3D que oferece uma interface gráfica e que permite aos consumidores visualizar e efectuar tarefas.

3.5.1. Infra-estrutura SOA

A infra-estrutura SOA foi implementada usando Java 2, Enterprise Edition (J2EE). Através desta Framework, um sistema de clusters providencia serviços críticos de modo a assegurar tempos mínimos de indisponibilidade e uma escalabilidade máxima. Por clusters, neste contexto entende-se um conjunto de servidores aplicacionais que de modo transparente executam a aplicação J2EE como se fosse uma única entidade.

Deste modo, o uso de clusters pode ser usado para correr uma aplicação de geoprocessamento em servidores paralelos. Mesmo que algum servidor falhe, a aplicação continuará acessível através de outros nós do cluster, garantindo-se também um tempo mínimo de indisponibilidade, se as

componentes do cluster estiverem redundantes. Também a performance desde pode ser melhorada com a adição de mais recursos para processamento.

Esta infra-estrutura SOA pode também ser vista quase como uma arquitectura de quatro camadas como ilustrado na Figura 9 onde work balancers podem distribuir a carga computacional pelos diversos servidores na camada.

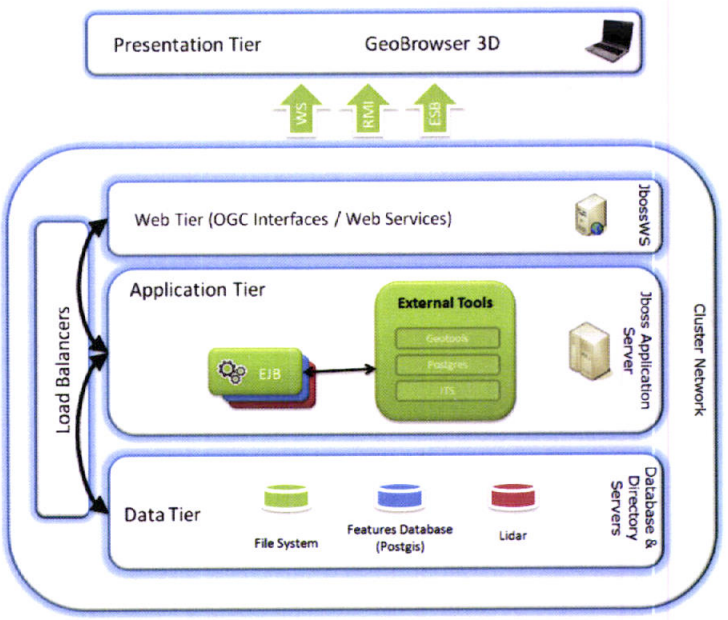

Figura 9 – Infra-estrutura SOA

A primeira camada da arquitectura é composta por um cliente 3D leve e acessível da web, que permite ao utilizador aceder, editar e gerir repositórios geográficos a partir de um ambiente interactivo 3D. Esta camada comunica apenas com a segunda camada – camada Web – de modo a aceder, por exemplo, aos serviços do OGC. Qualquer carga computacional necessária nesta camada, como o processamento de algoritmos, é efectuada numa terceira camada – camada Aplicacional – que faz uso da tecnologia EJB. Em casos em que a terceira camada necessita de aceder a qualquer tipo de dados, é então necessário que esta aceda a uma quarta camada – camada de dados.

3.5.1.1. Sub-Camada de Suporte

Nesta camada pretende-se fornecer funções base de suporte entre os consumidores e a infra-estrutura. Como tal, é possível encontrar aqui serviços de catálogos tais como o GeoNetwork, que permitem descobrir a aceder a serviços, resolvendo alguns dos cenários típicos em que geralmente os utilizadores não sabem onde encontrar os dados espaciais, ou quando tem que efectuar tarefas que pesquisa sobre informações que podem residir num numero de repositórios diferentes. Estas tarefas podem necessitar de muito tempo, e ainda se tornam mais críticas, por exemplo, em casos como o de uma planificação

urbana, que geralmente tem centenas de camadas diferentes, cada uma contendo uma variedade de dados geográficos provenientes de fontes de dados diferentes.

3.5.1.2. Sub-Camada de Serviços

Nesta camada é possível encontrar uma lista de serviços que podem ser acedidos e usados pelo utilizador. Alguns dos serviços que são considerados mais importantes e que foram implementados são: WMS, WFS e WPS.

Na Figura 10 é possível verificar como os diversos serviços se podem relacionar e comunicar entre si. Estes serviços podem também de modo transparente fazer referencia ou usar outros serviços externos.

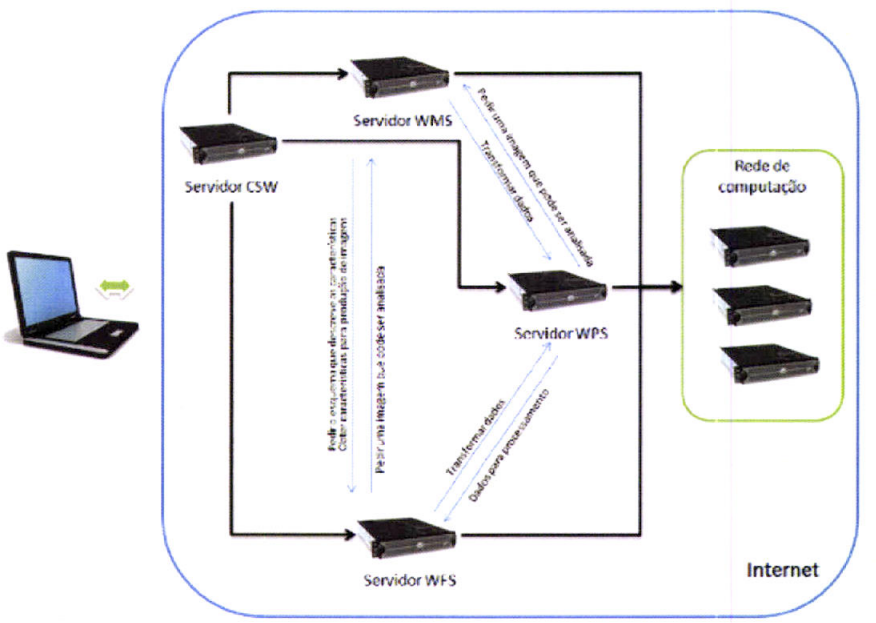

Figura 10 – Vista global da arquitectura

3.5.1.3. Sub-Camada Aplicações

Nesta camada está disponível um geobrowser 3D acessível através da tecnologia Java Web Start, e que permite explorar todas as capacidades desta arquitectura.

A tecnologia Java Web Start procura a portabilidade das applets, a sustentabilidade da tecnologia Servlets e JavaServer Pages (JSP), e a simplicidade de linguagens como XML e HTML. É uma aplicação Java-basead que permite executar e actualizar aplicações cliente de um servidor de Web standard. Ao lançar Java Web pela primeira vez, o utilizador carrega a aplicação cliente da Web; depois disso a aplicação ou pode ser iniciada por uma ligação em uma página de Web ou de ícones do Desktop ou menu Iniciar (Caso do Windows). Adicionalmente, e porque o Java Web Start é construído usando a tecnologia Java 2, herda a arquitectura de segurança da plataforma.

A arquitectura da aplicação disponibilizada está baseada no padrão arquitectónico Model-view-controller (MVC). No padrão arquitectónico MVC, o Model representa a informação (os dados) da aplicação, assim como os mecanismos para guardar e aceder; O View representa tudo o que diz respeito à visão (posição do utilizador, direcção, etc.); e o Controller responde aos processos e eventos (tipicamente acções do utilizador ou regras da aplicação), invocando indirectamente se necessário mudanças no Model ou no View.

A aplicação dispõe de alguns "serviços" internos primários:

- TaskService – gere as tarefas a serem executadas em simultâneo.
- RetrievalService – é um serviço genérico para a obtenção de dados da internet.
- Registry – gere as configurações do programa.
- Detector de plugins – Este serviço procura classes ou jars que estendam a interface Service e adiciona-o a lista de serviços disponíveis. Estes serviços têm acesso ao Controlador, e analogamente a toda a arquitectura, incluindo os outros serviços.

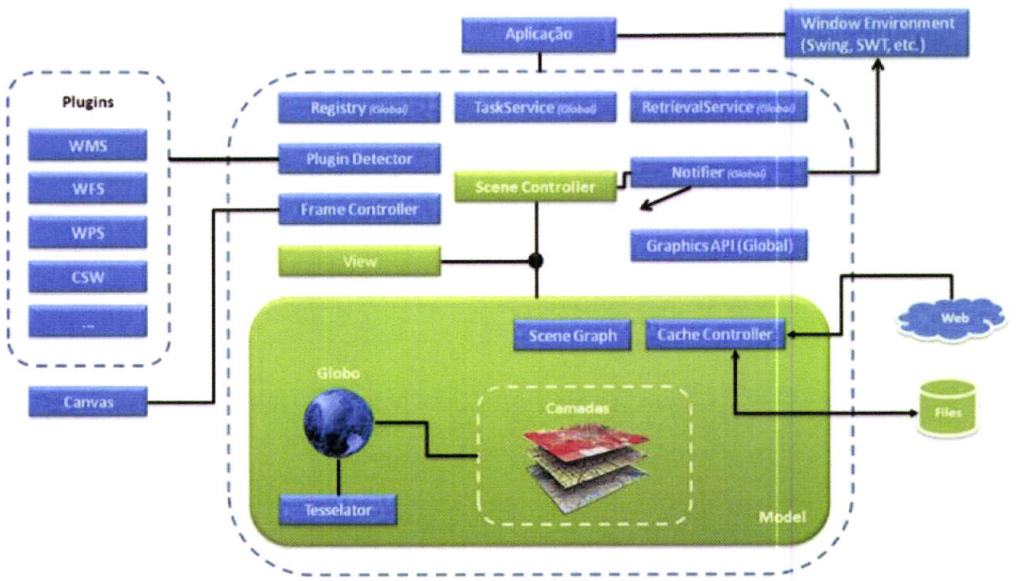

Figura 11 – Vista global da arquitectura do cliente

3.5.1.3.1. Task Service

Task Service gere as tarefas a serem executadas, de modo a que apenas um limite de tarefas especificado na configuração seja executado em simultâneo. Para além disso, controla os tempos de execução, e caso a tarefa demore mais que o tempo máximo especificado na configuração, a tarefa é automaticamente cancelada.

Deste modo a arquitectura tem um mecanismo que possibilita o controlo de todas a tarefas que irão ser executadas num ThreadPoolExecutor.

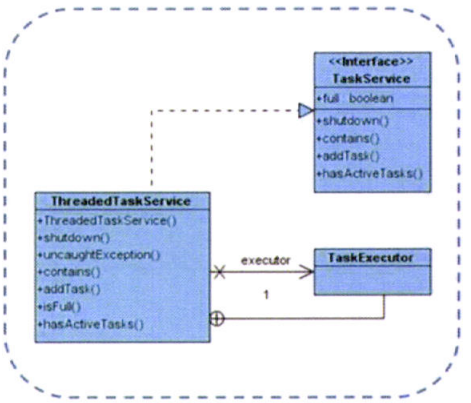

Figura 12 – Task Service

3.5.1.3.2. Retrieval Service

Retrieval Service é um serviço herdado da API World Wind que fornece mecanismos para obter recursos online.

Esta sub-arquitectura possibilita obter informações sobre os recursos que estão a ser descarregados da internet (como o tempo em que se começa a descarregar, o tamanho, etc.), possibilidade controlar o número de downloads simultâneos, e a prioridade de download.

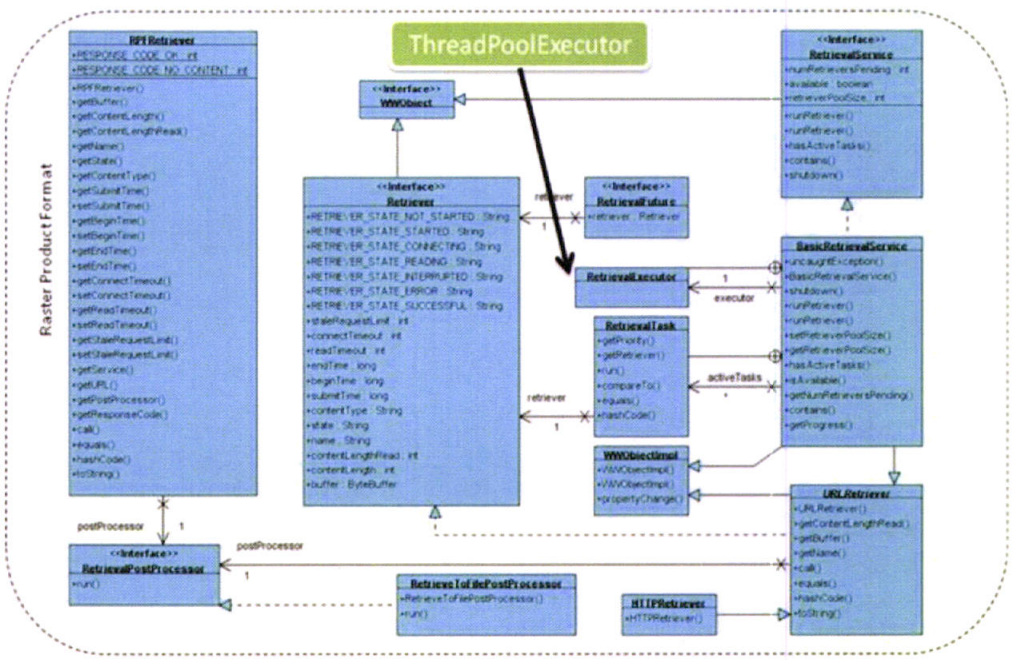

Figura 13 – Serviço de rede

Ao implementar a interface RetrievalPostProcessor é possível especificar o que fazer com dados depois de estarem disponíveis localmente. Todos os outros aspectos são geridos pela implementação por defeito da interface RetrievalService chamada BasicRetrievalService, e que cuida de todos os aspectos relacionados com a ligação à internet (proxy, disponibilidade, etc.).

A este nível foram implementadas algumas classes com URLRetriver, que simplesmente lê um URL e guarda os dados em disco, segundo o nome que lhe foi passado como argumento. Esta classe é usada para obter imagens de serviços WMS. Existem contudo outras derivações que permitem obter recursos apenas para a memória.

O funcionamento deste serviço foi alterado ligueirmanette de modo a ignorar pedidos que por alguma razão obtiveram um erro da parte do servidor. Deste modo não se gastam recursos a tentar obter algo que por algum motivo não é possível.

3.5.1.3.3. Registry

Este serviço gere todas as configurações do programa, que são inicialmente lidas de um ficheiro de propriedades. Este "serviço" faz uso da reflexão para instanciar as diversas classes da arquitectura, tornado deste modo fácil o uso de implementações diferentes, devendo estas apenas seguir a interface especificada.

Exemplo:

MyClass c = (MyClass)Class.*forName("it.mypack.MyClass").*newInstance();

3.5.1.3.4. Detector de Plugins

Este serviço funciona de modo análogo à reflexão, visto que usa uma abordagem similar para detectar se jars ou directórios contém alguma classe que implemente a interface Service.

Todas as classes que implementam a interface Service são adicionadas a uma lista de serviços.

A interface é composta por cinco funções getName(), getDirectory(), setSceneController(ISceneController), start() e stop().

Todos os serviços encontrados são adicionados no menu usando a estrutura de getDirectory()::String[] e o nome do serviço getName()::String, logo após este ser instanciado e definido o ISceneController.

Quando o utilizador clica no menu, é invocado o método start();

3.5.1.3.5. Rendering Loop

No que diz respeito ao rendering por parte do cliente, a cada frame são efectuadas oito etapas pela seguinte ordem:

1) A inicialização da frame em OpenGL

2) Aplicação da perspectiva e informação relativa ao ponto de vista
3) Criação do modelo de terreno (usando as partes que já existem em cache)
4) Limpeza da frame (glClearColor, glClear)
5) Fase de picking que inclui uma fase especial de render
6) Nova limpeza da frame (glClearColor, glClear)
7) Render da versão final da frame
8) São efectuadas operações de finalização (como glMatrixMode)

3.5.1.3.6. Picking

Para determinar o que está debaixo do rato, é usado um método de selecção que recorre ao uso de cores, já suportado pela API do Word Wind.

Durante a fase de selecção a arquitectura efectua o render da cena, de um modo simplificado atribuindo a cada objecto que se pode seleccionar uma cor única. Em casos de objecto complexos é desenhada uma caixa que incorpora o objecto. Depois de a cena ter sido renderizada, é lida a cor no z-buffer correspondente a posição (X,Y). Essa cor indica em O(1) o objecto seleccionado visto que é o índice do array (Cor – Cor inicial).

Este método oferece vantagens computacionais relativamente ao método geométrico, e permite seleccionar até 16.000.000 objectos diferentes, o que é suficiente tendo em conta as capacidades possíveis com o Java. É também coerente com o que o utilizador está a visualizar, mas só permite fazer picking do objecto no topo.

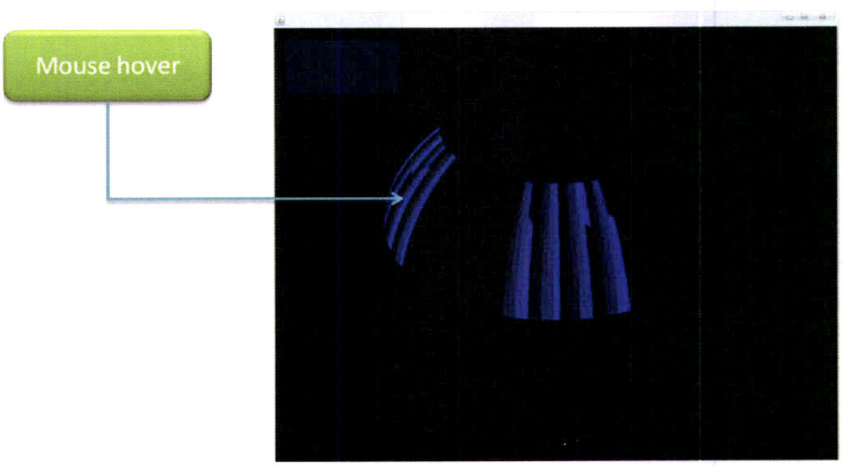

Figura 14 – Mecanismo de Selecção

Na Figura 14 é possível ver uma imagem resultante da fase de picking. Apenas o sector do globo que está seleccionado como o rato é renderizado (incluindo toda a informação incluída nele), assim como também o sector que está em frente a câmara, pois é também na fase de picking que se define a prioridade para as imagens a obter da internet (dando prioridade as que estão seleccionadas e as que estão em frente ao utilizador).

3.5.1.3.7. Camada

Todas as camadas são geridas por um gestor de camadas. Este gestor organiza as camadas por tipo e define a sua ordem de rendering. É também responsável pela criação e actualização de uma lista com todas as camadas visíveis, necessária à API do World Wind, e que é renderizada a cada frame.

As camadas são em geral um mecanismo para obter a textura de determinado sector da terra. Ao ser pedida a textura por parte do motor de rendering, a camada procura a textura na memória (disponível quando usada recentemente), em seguida no disco. Caso não esteja disponível em nenhum dos sítios elabora um pedido ao respectivo servidor online.

Quando é criada uma camada na arquitectura deve ser definida pelo menos a dimensão da imagem, o local relativo na cache, o nome do dataset no servidor, o endereço do servidor, formato de imagem, Bounding Box (de modo a filtrar as camadas visíveis ao utilizador) e como construir o pedido. Adicionalmente mais parâmetros podem ser definidos tais como informação relativa ao número de níveis (o numero de níveis especifica o numero de intervalos de altitude que usam a mesma imagem), etc.

O exemplo seguinte ilustra como definir as propriedades:

```
@Override
private static LevelSet makeLevels(String layerName) {
    AVList params = new AVListImpl();
    params.setValue(AVKey.TILE_WIDTH, 256);
    params.setValue(AVKey.TILE_HEIGHT, 256);
    params.setValue(AVKey.DATA_CACHE_NAME, "Directorio/"+ layerName);
    params.setValue(AVKey.DATASET_NAME, layerName);
    params.setValue(AVKey.SERVICE, "www.servidor.com");
    params.setValue(AVKey.FORMAT_SUFFIX, ".png");
    params.setValue(AVKey.SECTOR, Sector.FULL_SPHERE);
    params.setValue(AVKey.TILE_URL_BUILDER, new URLBuilder());
    ...
    return new LevelSet(params);
}
```

Um exemplo de um construtor básico e estático de um pedido WMS poderia ser:

```
private static class URLBuilder implements TileUrlBuilder {
    public URL getURL(Tile tile, String imageFormat) throws  MalformedURLException {
        Sector s = tile.getSector();
        String sb tile.getLevel().getService() +
         "version=1.3.0"+
```

```
            "&request=GetMap" +
            "&layers="+ tile.getLevel().getDataset() + "crs=EPSG:4326" +
            "&width=" + tile.getLevel().getTileWidth() +
            "&height=" + tile.getLevel().getTileHeight() +
            "&bbox=" + s.getMinLongitude().getDegrees() + "," + s.getMinLatitude().getDegrees() + "," +
                        s.getMaxLongitude().getDegrees()  + "," + s.getMaxLatitude().getDegrees() +
            "&format=image/png&SERVICE=WMS&bgcolor=0x000000&transparent=TRUE";
            return new URL(sb);
        }
    }
```

Neste caso o cliente apenas suporta a versão 1.3.0 assim como a projecção EPSG:4326 (Latitude/Longitude), contudo em futuras implementações poderia-se estender a outras projecções visto que apenas é necessário reprojectar o parâmetro BBOX.

3.6. EJB e Serviços Web

A adopção da tecnologia Enterprise JavaBeans (EJB) permite criar um modelo elegante de componentes, independente de qualquer especificação de plataforma, protocolo proprietário ou infra-estrutura de middleware. Pode também ser usada para transacções de negócios de pequena e larga escala e executar aplicações distribuídas e transaccionais de modo seguro.

Quando os requerimentos para o geoprocessamento aumentam, as EJB podem ser migradas para outros ambientes operativos mais poderosos sem quaisquer dificuldades ou alterações adicionais.

Na implementação desta tese foram usados dois dos três tipos de beans: Session beans e Message Driven beans. Stateless session beans são usadas para mapping e geoprocessamento e message-driven beans para processamento e notificação assíncrono (ver item 3.7 abaixo).

3.7. Suporte para transacções que exigem espera

Este sistema está também preparado para transacções longas, permitindo assim evitar alguns dos possíveis problemas relacionados com a rede, como por exemplo o caso em que um cliente é desconectado no meio de uma transacção, na qual apenas estava a espera da resposta. Num caso normal, quando o cliente se re-conecta deixa de ter a hipótese de reaver a informação sobre a conclusão da operação em curso, que na maioria dos casos é cancelada.

Quando uma tarefa leva algum tempo a ser processada, então é adicionada a uma lista de espera como apresentado na Figura 15. Na resposta imediata à operação o cliente recebe um documento XML (ver operação Execute do WPS), que permite ao cliente ver a qualquer momento o estado da operação. No caso em que o cliente suporta o sistema Publisher/Subscriber, quando a tarefa é concluída o servidor publica uma mensagem num tópico, que é recebida pelos clientes subscritos. Esta notificação é também

importante, porque deste modo permite a todos os clientes conectados saber quando os dados são alterados, permitindo assim actualizar a informação que está a ser apresentada ao utilizador.

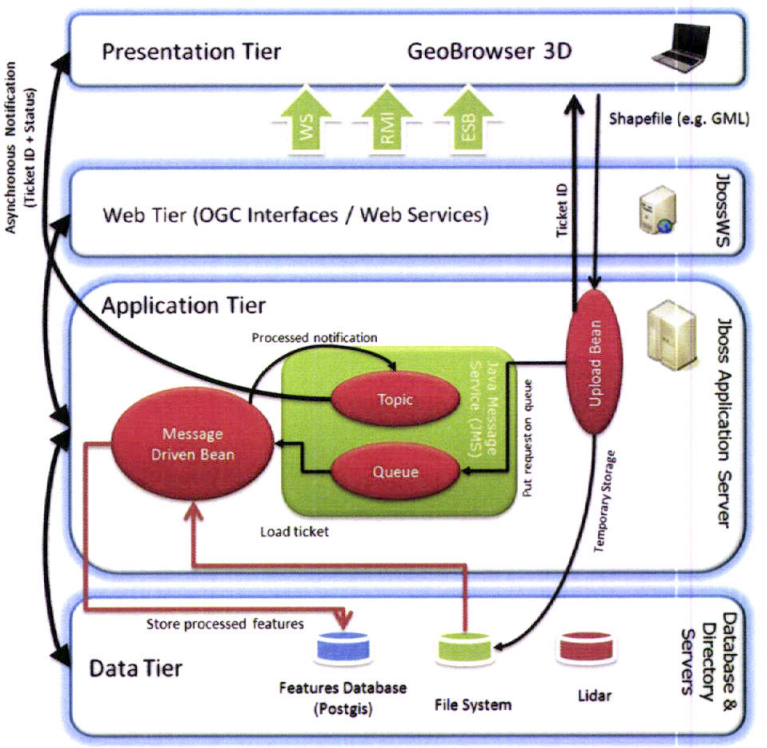

Figura 15 – Sistema de mensagens assíncronas

Um exemplo do uso desta abordagem é quando se pretende efectuar upload de ficheiros que contem dados geoespaciais. O cliente efectua o upload do ficheiro, que vai para uma fila de espera pois antes de ser armazenado pode necessitar de pré-processamento como por exemplo ser reprojectado. Menssage-Driven beans vão processando os pedidos que vão chegando e assim que terminam enviam a mensagem para o tópico, que é depois recebida pelos clientes de modo assíncrono. As MDBs vão também actualizando o ficheiro de estado disponível online, e que pode ser usado por clientes que não suportam o sistema Publisher/Subscriber, de modo especificado pelo padrão WPS. Esta operação de adição de novos dados ao servidor é efectuada através do protocolo WPS, onde todos os dados necessários são passados como argumentos. Os dados incluem por exemplo autenticação, assim como outros parâmetros específicos desta operação.

3.8. Serviços Implementados

3.8.1. Serviço WMS

Para a implementação do serviço Web Map Service (WMS) partiu-se do código do servidor de WMS que a NASA disponibiliza para teste de algumas das suas aplicações. O software foi escolhido devido ao facto de todo este ser baseado em interfaces e de conter o suporte necessário para a manipulação de imagens pré-processadas em disco. É importante referir que contudo este software não suporta dados vectoriais, quer isto dizer que apenas é capaz de responder a pedidos com imagens pré-processadas e existentes em disco.

Originalmente, ao iniciar, o serviço lia do ficheiro de configurações a classe que instanciava cada camada disponível, assim como os parâmetros necessários para a sua inicialização (nome da camada, fonte de dados, etc.). Esta classe era responsável pelo seu processamento (MapGenarator). Deste modo se o serviço disponibiliza-se 10000 camadas, o ficheiro de configuração deveria conter a informação necessária para cada uma das 10000 camadas. Este tipo de abordagem 1:1 para as configurações é usado por todos os servidores de WMS open source disponíveis.

Visto que o servidor de teste disponibiliza algumas centenas de camadas, e que também se pretendia que estas pudessem ser adicionadas e removidas dinamicamente, sem a necessidade de rescrever o ficheiro de configurações, foi implementada uma nova arquitectura para o serviço, como ilustrada na Figura 16.

Ao código inicial foi adicionado um novo conceito – MapSource – e à arquitectura foram adicionadas mais duas camadas (camada aplicacional e camada de dados) que permitem assim distribuir o work load e os dados por um cluster, melhorando a tolerância a erros, visto que assim os dados podem estar redundantes em vários nós do cluster, e melhorando a performance, visto que os dados e a carga computacional podem ser distribuídos. O sistema torna-se deste modo também escalável, bastando para isso aumentar o número ou a qualidade dos recursos.

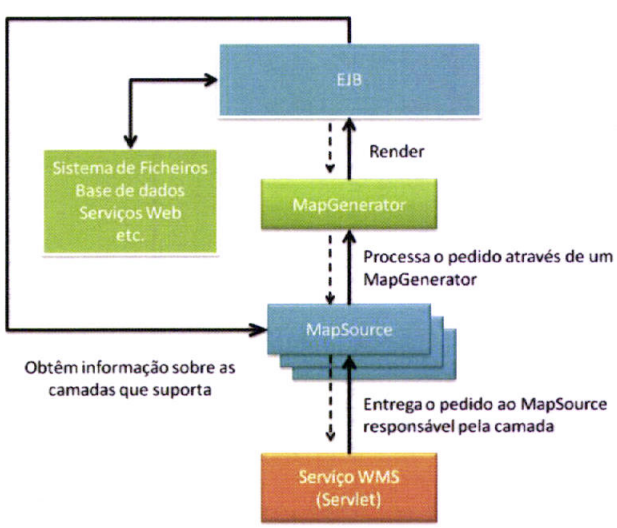

Figura 16 – Arquitectura do serviço WMS

Nesta nova arquitectura o ficheiro de configuração define que classes (que estendem MapSource) devem ser inicializadas, e os parâmetros que estas podem necessitar tais como (quantidade de memoria máxima a usar, numero de conexões, etc.). Quando o serviço gera o catálogo, em vez de usar as informações relativas a cada MapGenerator (1:1), é usada a lista de camadas devolvida pelo MapSource (1:n), onde n pode ser apenas uma camada.

Ao ser pedida uma camada, o serviço verifica que MapSource a disponibiliza, e entrega-lhe o pedido. Nesta arquitectura a interface MapSource é apenas responsável pela implementação dos mecanismos que permitem descobrir que camadas estão disponíveis (que por exemplo podem estar listadas numa tabela SQL, serem provenientes de uma base de dados, estarem listadas num ficheiro de texto, definidas na implementação, etc.), e deve ser implementada conforme as diversas necessidades. Esta classe deve também implementar outra interface MapGenerator que é responsável por descodificar um pedido WMS e criar uma resposta adequada.

Visto que o servidor original da NASA não vem com suporte para rendering de dados vectoriais em run-time, foi implementado um MapGenerator assim como um módulo para descodificar os pedidos de WMS.

De forma a dar continuidade a ideia de escalabilidade, até porque agora é também esperado o processamento de dados vectoriais em tempo real qualquer necessidade de rendering é efectuada pela segunda camada (camada aplicacional), que acede a uma terceira camada (camada de dados), que pode também estar disponível em diversos computadores.

No que diz respeito a comunicação, as transformações XML-Java e Java-XML podem ser obtidas através de bindings da especificação, como o ilustrado no exemplo que se segue.

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
  "format",
  "onlineResource"
})
@XmlRootElement(name = "LegendURL")
public class LegendURL {

  @XmlElement(name = "Format", required = true)
  protected String format;

  @XmlElement(name = "OnlineResource", required = true)
  protected OnlineResource onlineResource;

  @XmlAttribute
  protected BigInteger height;

  @XmlAttribute
  protected BigInteger width;

  /**
   * Gets the value of the format property.
   */
  public String getFormat() {
    return format;
  }

  /**
   * Sets the value of the format property.
   */
```

```
    public void setFormat(String value) {
        this.format = value;
    }

    /**
     * Gets the value of the onlineResource property.
     */
    public OnlineResource getOnlineResource() {
        return onlineResource;
    }

    /**
     * Sets the value of the onlineResource property.
     */
    public void setOnlineResource(OnlineResource value) {
        this.onlineResource = value;
    }

    /**
     * Gets the value of the height property.
     */
    public BigInteger getHeight() {
        return height;
    }

    /**
     * Sets the value of the height property.
     */
    public void setHeight(BigInteger value) {
        this.height = value;
    }

    /**
     * Gets the value of the width property.
     */
    public BigInteger getWidth() {
        return width;
    }

    /**
     * Sets the value of the width property.
     */
    public void setWidth(BigInteger value) {
        this.width = value;
    }

}
```

De modo análogo foram implementados os binding para todos os objectos referidos na descrição do protocolo no Anexo II. Depois os objectos são codificados em XML usando o excerto de código seguinte:

```
JAXBContext jaxbContext = JAXBContext.newInstance(Capabilities.class);

Marshaller marshaller = jaxbContext.createMarshaller();

marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

marshaller.setProperty(Marshaller.JAXB_FRAGMENT, true);


ByteArrayOutputStream stream = new ByteArrayOutputStream();

marshaller.marshal(capabilities, stream);

xml += stream.toString();
```

Para converter o XML para objectos Java o cliente efectua o unmarshaller através do código:

```
JAXBContext jaxbContext = JAXBContext.newInstance(Capabilities.class);
Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();
Object r = unmarshaller.unmarshal(new ByteArrayInputStream(xml.getBytes("UTF-8")));
if(r instanceof Capabilities)
        this.capabilities = (Capabilities)r;
```

Neste serviço WMS foram também implementadas as alterações referidas no capítulo anterior que tornam possíveis o pedido de imagens delimitadas por uma geometria.

De modo a criar o resultado são efectuados os seguintes passos:

Primeiro é gerada uma imagem (mask) com a forma desejada como filtro como ilustrado no exemplo que se segue, através de um polígono que tem o aspecto de uma circunferência. Esta imagem é gerada do mesmo modo que é renderizada uma característica. A diferença é que a característica é dada pelo utilizador como parâmetro e renderizada com um estilo preto.

Figura 17 – Mascara a aplicar ao mapa

Depois é gerada a imagem usado o procedimento normal. O resultado é por fim interceptado com a mascara, usando o seguinte código:

```
int w = image.getWidth();
int h = image.getHeight();
BufferedImage outImage = new BufferedImage(w, h, BufferedImage.TYPE_INT_ARGB);
int color = Color.black.getRGB();
for (int y=0; y<h; y++) {
    for (int x=0; x<w; x++) {
        int valueMask = imageMask.getRGB(x,y);
        int value = image.getRGB(x,y);
        if (valueMask == color) outImage.setRGB(x,y,value);
```

```
    }
}
```

A Figura 18 representa um exemplo de um resultado a um pedido WMS que especifica nos parâmetros, o fundo como transparente, o tipo de imagem como PNG, e uma BBOX correspondente à geometria apresentada na mascara anterior (ver Figura 17).

Figura 18 – Resultado da imagem combinada com a mascara

3.8.1.1. Sistema de cache

É também importante, e não muito comum nas implementações disponíveis em Open Source, o uso de cache para o caso das camadas que são geradas de dados vectoriais em run-time.

Se não existir um sistema de cache, o sistema deverá de ser capaz de em run-time gerar e entregar todas as imagens necessárias à navegação por parte das aplicações clientes. Apesar de este impacto ser amortizado com uma cache de primeiro nível no cliente, uma cache ao nível do servidor permite que este apenas gaste recursos computacionais uma vez, podendo depois entregar a mesma imagem, existente em cache a múltiplos clientes.

De modo a assegurar a consistência na cache foi implementada uma bean intermediária, que intercepta pedidos de rendering, verificando primeiro se estes existem em cache. Como as características são controladas através de uma versão, essa versão é usada juntamente com o identificador da camada e com o identificador do estilo, e outros parâmetros que afectam a imagem produzida, como as dimensões e cor de fundo, para guardar a imagem em cache. Quando o estilo de uma camada é modificado, o controlador de cache recebe um pedido para eliminar uma camada, e procede a limpeza da cache. Toda esta sub-arquitectura é ilustrada na imagem Figura 19.

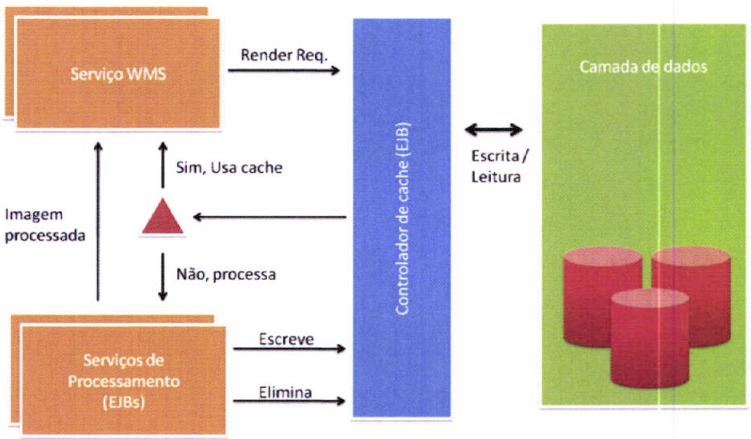

Figura 19 – Arquitectura do serviço WMS

O serviço de WMS com estes dois níveis de cache passou a apresentar desempenhos superiores a 40% relativamente a outros pequenos servidores de WMS que efectuam rendering em run-time de dados dinâmicos.

3.8.2. Serviço WFS

Para o serviço WFS foi usado o GeoServer, que é um dos servidores que mais rapidamente evolui, e o que suporta mais funcionalidades relativamente ao serviço WFS.

Cada vez que se adiciona ou remove alguma camada é necessário actualizar a lista de camadas existentes, reescrevendo o ficheiro de configurações que suporta (base de dados e ficheiros) e recarregá-lo novamente. É fácil identificar alguns problemas nesta arquitectura. Primeiro salta a camada aplicacional, e quer isto dizer, que não possibilita a existência de operações intermediárias tais como por exemplo aceder de modo distribuído a dados da camada de dados e antes de os entregar ao cliente reprojecta-los ou efectuar qualquer outro tipo de processamento usando por exemplo, o serviço WPS.

O problema existe porque o GeoServer cria uma classe que lê a camada da fonte especificada, contudo devido ao facto de não ser uma implementação modular, isto é, que não segue por exemplo um conjunto de interfaces, foi impossível em tempo útil proceder a sua alteração. Contudo, elementos da equipa de desenvolvimento do geoserver parecem empenhados em pelo menos tornar a classe que gera os dados para cada camada mais modular (classe que acede aos dados em base de dados ou ficheiros). Nessa altura será então possível criar uma classe que de modo semelhante ao serviço WMS obtém os dados da camada aplicacional, onde poderá ser possível introduzir novas optimizações ao sistema, através de mecanismos de cache e processamento de dados.

Contudo persistirá o problema de ter que reescrever o sistema de configuração do GeoServer em todos os nós do cluster, visto que continuará a não ser possível integrar de modo modular uma classe responsável pela descoberta das camadas disponíveis.

3.8.3. Serviço WPS

O uso de uma infra-estrutura cliente-servidor é claramente favorável a actividades de processamento. Deste modo actividades de processamento pesadas passam a estar acessíveis a um público mais amplo, devido ao facto da se transferir a carga computacional de processamento para redes distribuídas de processamento tais como clusters, mantendo deste modo o cliente leve e pequeno, quase limitado a um visualizador que apenas apresenta imagens e dados processados.

O serviço WPS incorporado nesta arquitectura contém o porte de quase todos os algoritmos (+200) de uma das ferramentas mais faladas na área do geoprocessamento - SEXTANTE.

Este serviço permite ao cliente efectuar qualquer transformação, implementada através de processos, assim como obter os metadados associados a cada processo, que incluem por exemplo descrições sobre as suas entradas e saídas de dados.

Actualmente o serviço suporta a exposição dos processos via GET, POST, e SOAP, permitindo ao cliente escolher o mecanismo de interface mais apropriado. Os processos disponíveis na implementação ficam ao critério do proprietário do serviço, e apesar da implementação WPS ter sido definida para dados georreferenciados, pode ser usada para qualquer outro tipo de dados.

A Framework 52N WPS foi a implementação de escolha neste caso devido ao facto de já permitir expor funcionalidades OGC- e W3C- compilantes e por ser a mais funcional de todas as encontradas.

52N WPS, como implementação do padrão OGC WPS, permite aos programadores estender a instância WPS com a adição de novos processos, seguindo a estrutura ilustrada na Figura 20.

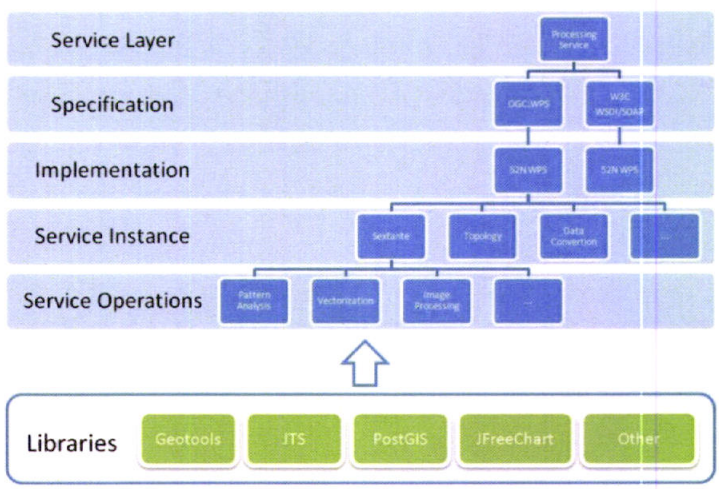

Figura 20 – Composição do serviço WPS

A Framework 52N WPS foi estendida a uma arquitectura de três camadas (ver imagem Figura 21). Com esta nova arquitectura foi possível ver aumentos de velocidade no processamento de alguns algoritmos superiores aos 1000%. Um exemplo disso é o cálculo do perfil de um terreno, que passou a ser calculado simultaneamente em múltiplos computadores, responsáveis por diversas secções do terreno. Deste

modo o processamento dos algoritmos é efectuado de modo transparente pelo cluster de processamento.

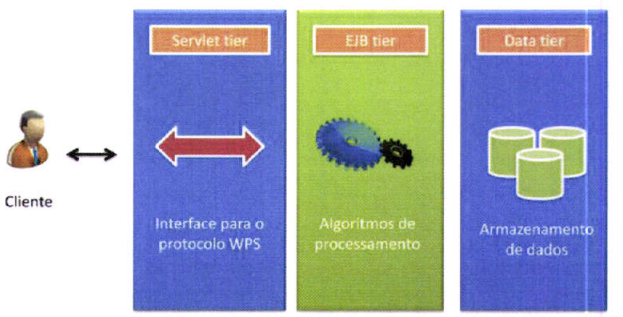

Figura 21 – Arquitectura do serviço WPS

3.9. Serviços suportados pela aplicação

Na interface desenvolvida o utilizador pode adicionar endereços de serviços ou de catálogos web (CS-W) que deseja usar, contudo actualmente apenas existem plugins para WMS, WFS-T e WPS. Depois os serviços (ou acções adicionadas ao menu pelos plugins) podem ser acedidos de um menu 3D que é transparentemente populado (veja a imagem da esquerda na Figura 22). O tipo e nome são obtidos automaticamente dos metadados do serviço caso o botão referencie um serviço OGC. Quando uma opção é seleccionada no menu, um novo anel exterior é activado fornecendo o acesso a opções adicionais relacionadas com a opção.

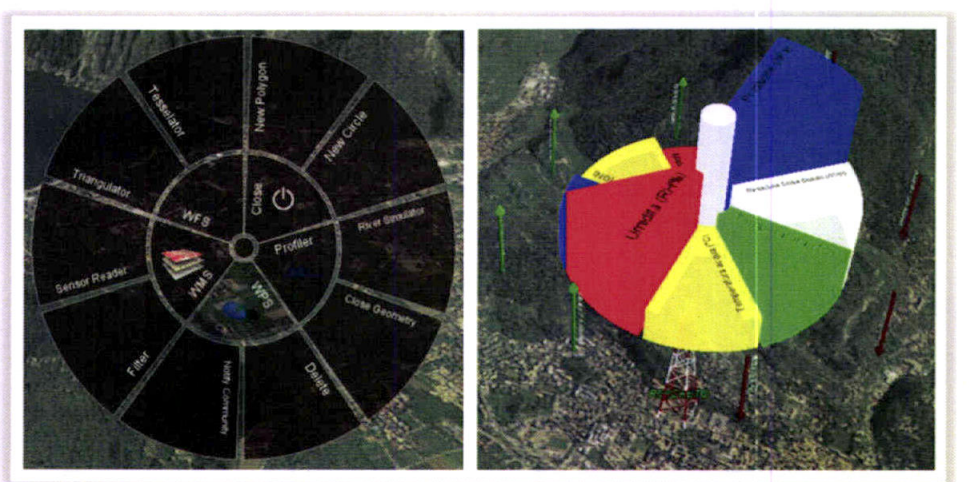

Figura 22 – Interface do Menu 3D

Depois de o utilizador seleccionar uma opção, o menu é redimensionado e movimenta-se para o canto inferior esquerdo de modo a libertar espaço. Quando o utilizador clica no menu, este volta ao centro da

tela e passa ao tamanho normal. Deste modo nunca se perde o estado anterior, como acontece com os menus comuns, onde depois de se seleccionar uma opção o menu volta ao estado inicial (desaparece). É necessário manter o estado, principalmente no caso de encandeamento de processos, em que é necessário aceder múltiplas vezes ao menu. Esta abordagem reduz assim o tempo necessário para atingir um objectivo.

A interface 3D fica particularmente útil ao lidar com datasets numéricos, por exemplo, quando o utilizador precisa aceder em tempo real a dados de um sensor localizado no terreno. Estes sensores podem conter informações sobre barómetro, humidade, visibilidade, direcção de vento, etc. que são obtidas em tempo real de serviços web. Neste caso a interface 3D é usada para representar a informação do respectivo dataset, como ilustrado na imagem da direita na Figura 22. No exemplo, a altura do menu 3D é usada para representar os últimos dados disponíveis, em comparação com o mínimo e o máximo, e a seta indica a tendência da variação (aumentando / verde ou diminuindo / vermelho). O valor de variação é indicado junto à seta. Quando o utilizador clica no menu é fornecida mais informação (veja Figura 23).

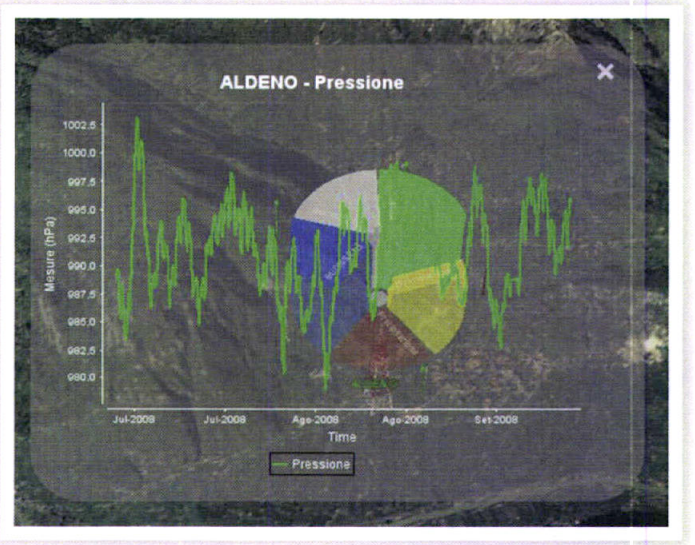

Figura 23 – Gráfico dos valores do sensor gerado em run-time

Com esta abordagem é possível relacionar diversos campos (por exemplo de um sensor). Para um especialista no tempo, o exemplo apresentado pode ser de extrema utilidade, pois possivelmente ao ver que a temperatura está a descer e a humidade a aumentar (e outros factores), pode facilmente tirar conclusões avançadas sobre o que está ou vai acontecer (e.g. se está ou se vai chover).

3.9.1. Serviço WMS

A aplicação permite também ao utilizador através de uma janela comum, escolher o estilo para uma camada. Eventualmente e caso o utilizador tenha conhecimentos de programação, pode programar o

SLD directamente. Depois o estilo é enviado ao servidor através da operação PutStyle, e fica automaticamente disponível a todos os utilizadores.

Os utilizadores podem saber se o estilo foi alterado através de dois métodos. Imediatamente após a sua alteração se subscritos num serviço Publisher/Subscriber, ou através da comparação da versão da camada.

Quando um estilo é modificado, a aplicação cliente deve remover toda a cache desactualizada.

Através do vídeo "Interface WMS.avi" presente anexo, é possível visualizar todo este processo que vai desde a adição de uma nova camada a sucessivas alterações no seu estilo.

3.9.2. Serviço WFS-T

Quando o cliente está em modo de edição, ao ser seleccionada uma camada, são obtidas do servidor as características relativas à camada com uma fechadura. No caso de não ser possível obter uma fechadura é apresentada uma mensagem informativa ao utilizador. Depois e caso seja possível obter a fechadura, são verificadas as propriedades, pois é necessário ter em consideração o tipo de geometria (Por exemplo uma Polyline tem a mesma forma que um Polygon mas não apresenta preenchimento), caso contrário os resultados poderão não ser correctos quando se aplica o estilo. O cliente obtém o respectivo estilo (através da operação GetStyle do serviço WMS) para a camada e faz a sua aplicação. É de notar que agora o cliente está a trabalhar num modo vectorial, e devido a isso existe uma perca de performance quando se trabalha com grandes dataset.

Como ilustrado na Figura 24, pode-se aceder às diversas funcionalidades do WFS-T através do menu inferior ou através do uso de um menu contextual.

A selecção das características pode ser efectuada, através do click sobre ela caso seja fácil, ou no caso em que exista um grande aglomerado de informação e existam características sobrepostas através de uma funcionalidade que ao clicar numa posição devolve todas as características que estejam próximas, devidamente relacionadas num grafo, e que podem ser seleccionadas deste.

Figura 24 – Interface WFS-T

Esta interface permite para além da visualização e da manipulação, a criação de geometrias 2D de modo intuitivo. De modo a auxiliar a criação de novas características, podem ser usados métodos básicos como por exemplo o decalco de ortofotos, onde é apenas necessário apresentar a ortofoto e depois com o rato ir definindo os pontos.

Para a elaboração deste plugin partiu-se do módulo WFS do GeoTools, actualmente classificado com unsupported visto que para além de não ser suportado por programadores está praticamente infuncional.

Uma vantagem de partir deste módulo relativamente a muitos outros é a sua modularidade, e a facilidade com que se pode relacionar com as outras interfaces da biblioteca. Como consequência, a única coisa necessária é implementar como efectuar um pedido e interagir usando protocolo WFS descrito em 0 abaixo, visto que GeoTools já fornece suporte para gerar e transformar GML para um modelo de características.

Este módulo de WFS implementa assim a interface do GeoTools DataStore:

```
public abstract interface org.geotools.data.DataStore extends org.geotools.data.DataAccess {
    public abstract void updateSchema(String arg0, SimpleFeatureType arg1) throws IOException;
    public abstract String[] getTypeNames() throws IOException;
```

```
    public abstract SimpleFeatureType getSchema(String arg0) throws IOException;
    public abstract FeatureSource getView(Query arg0) throws IOException, SchemaException;
    public abstract FeatureSource getFeatureSource(String arg0) throws java.io.IOException;
    public abstract FeatureReader getFeatureReader(Query arg0, Transaction arg1) throws IOException;
    public abstract FeatureWriter getFeatureWriter(String arg0, Filter arg1, Transaction arg2) throws IOException;
    public abstract FeatureWriter getFeatureWriter(String arg0, Transaction arg1) throws IOException;
    public abstract FeatureWriter getFeatureWriterAppend(String arg0, Transaction arg1) throws IOException;
    public abstract LockingManager getLockingManager();
}
```

Agora para aceder a dados de um servidor WFS é exactamente o mesmo que aceder a dados num ficheiro Shapefile ou outra base de dados qualquer.

```
/** Step 1 - connection parameters */
Map<String, Object> connectionParameters = new HashMap<String, Object>();

String url = server.getServiceURL() + "&REQUEST=GetCapabilities&VERSION=" + server.getProtocolVersion();
connectionParameters.put("WFSDataStoreFactory:GET_CAPABILITIES_URL", url);
connectionParameters.put("WFSDataStoreFactory:USERNAME", server.getUsername());
connectionParameters.put("WFSDataStoreFactory:PASSWORD", server.getPassword());

/** Step 2 - connection */
datastore = DataStoreFinder.getDataStore(connectionParameters);

/**
 * Step 3 – discovery
 * Através dos parâmetros GeoTools instancia a respectiva interface, se compatível com alguma das disponíveis
 */

typeNames = datastore.getTypeNames();
```

De modo a inserir dados no servidor, e considerando que se tem permissões é necessário criar uma nova transacção (Transaction da API geotools) com os dados necessários à autenticação. É importante referir que é criada uma fechadura quando se começa a editar alguma característica no cliente, e que é removida assim que o cliente termina o modo de edição.

De modo a inserir características numa camada é usado o código que se segue.

```
public void insert(String layer, FeatureCollection<SimpleFeatureType, SimpleFeature> fc, Transaction transaction) {
        try {
```

```
                    FeatureStore<SimpleFeatureType, SimpleFeature> store =
                            (FeatureStore<SimpleFeatureType, SimpleFeature>) datastore.getFeatureSource(layer);
                    store.addFeatures(fc);
                    transaction.commit();
            } catch(Exception e){
                    e.printStackTrace();
                    transaction.rollback();
            }
}
```

A lista de características é convertida para GML e enviada ao servidor num documento de GML semelhante ao ilustrado em 0 abaixo.

É de reparar que neste excerto de código apenas é possível inserir características de um tipo, isto deve se ao facto de não ser necessário para este cliente inserir características de diversas camadas ao mesmo tempo, visto que o utilizador apenas pode trabalhar com uma de cada vez (limitação na aplicação implementada). Contudo basta inserir novas características em outras camadas antes de efectuar o commit().

De modo análogo é efectuada a remoção:

```
public void remove(String layer, Filter filter,  Transaction transaction) {
        try {
                FeatureStore<SimpleFeatureType, SimpleFeature> store =
                        (FeatureStore<SimpleFeatureType, SimpleFeature>) datastore.getFeatureSource(layer);

                store.removeFeatures( filter );
                transaction.commit();
         } catch(Exception eek){
                eek.printStackTrace();
                transaction.rollback();
        }
}
```

E a actualização, que neste caso e pelas mesmas razões que a inserção apenas permite actualizar um atributo de cada vez.

```
public void update(String layer, Filter filter, String attributeName, Object value, Transaction transaction) {

        FeatureStore<SimpleFeatureType, SimpleFeature> store;
        try {
```

```
                store = (FeatureStore<SimpleFeatureType, SimpleFeature>) (datastore).getFeatureSource(layer);
                store.setTransaction(transaction);

                SimpleFeatureType featureType = store.getSchema();
                AttributeDescriptor attribute = featureType.getDescriptor(attributeName);
                try {
                        store.modifyFeatures(attribute, value, filter);
                        transaction.commit();
                } catch (Exception eek) {
                        eek.printStackTrace();
                        transaction.rollback();
                }
        }
        catch (Exception e1) {
                e1.printStackTrace();
        }
}
```

Para o caso da remoção e da actualização, os filtros podem ser construídos invocando a seguinte função, que faz uso da biblioteca GeoTools. Filter é uma classe que produz o código XML para o filtro usando a especificação (Vretanos, 2005).

```
public Filter getFilter(String [] ids, boolean or){
        FilterFactory ff = CommonFactoryFinder.getFilterFactory( GeoTools.getDefaultHints() );
        List<Filter> filters = new ArrayList<Filter>();
        for(String id : ids)
                filters.add(ff.id( Collections.singleton( ff.featureId(id))));
        return (or) ? ff.or(filters) : ff.and(filters);
}
```

Para uma melhor visualização da interacção veja o ficheiro em anexo "Interface WFS-T.avi".

3.9.3. Serviço WPS

Através do uso dos processos disponíveis em serviços web, um utilizador pode automatizar tarefas SIG que podem ser bastante diversas, tais como descobrir caminhos mínimos entre dois pontos, sítios onde a criminalidade é mais elevada, predição das consequências de uma inundação ou de uma tempestade através de transformação de dados, análise de imagens, etc.

No momento em que o utilizador selecciona um processo do menu, a aplicação cliente irá pedir ao respectivo servidor toda a informação sobre o processo, que incluirá todos os parâmetros necessários

assim como informação sobre estes, tais como o tipo e restrições. Para mais informações sobre as informações devolvidas veja o Anexo IV, item 1.2 abaixo.

Usando as informações obtidas da resposta do servidor, é criado um conjunto de objectos 3D na posição assinalada pelo utilizador como ilustrado no exemplo da Figura 25. Estes objectos 3D são uma interface simples, que permite usar as funcionalidades do protocolo WPS de modo transparente, permitindo por exemplo, que o utilizador obtenha toda a informação disponibilizada sobre o processo através de um simples click no botão de ajuda.

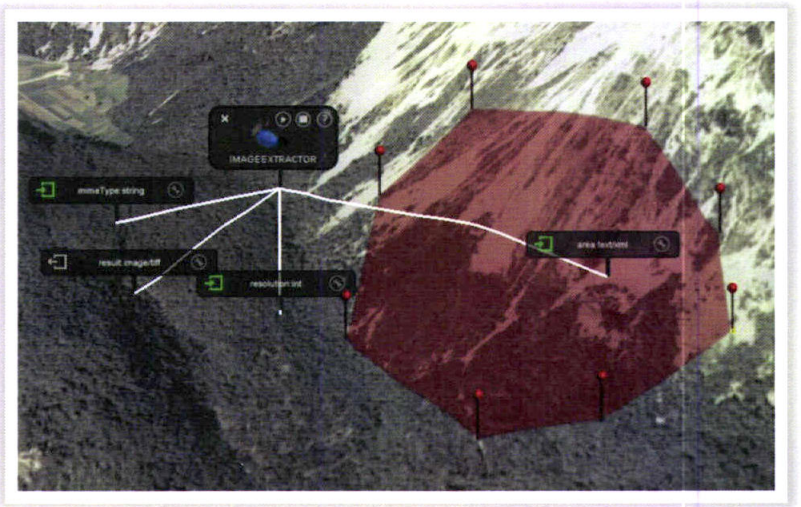

Figura 25 – Representação de um processo

Cada processo representa uma caixa negra no que diz respeito a como as coisas são processadas, onde apenas é requerido ao utilizador que defina os devidos parâmetros de entrada. Deste modo quando se pretende processar algo relativo a determinada área geográfica, basta seleccionar a área de interesse e ligar o respectivo parâmetro de entrada do processo a essa mesma, numa abordagem totalmente visual, quebrando assim os métodos tradicionais em que é necessário saber o nome, um código ou as coordenadas geográficas da área de interesse.

Ao executar o processo (usando o respectivo icon), é inicializada uma comunicação de um modo interoperável com respectivo serviço web, onde são passados os diversos parâmetros de entrada. A resposta do servidor deve estar tipada de acordo com o tipo de dados esperados para o resultado, aplicando se o mesmo aos parâmetros de entrada. Caso contrário a aplicação apresentará uma mensagem de erro.

A Figura 26 ilustra como são representados os parâmetros de entrada e de saída de um processo. Ao centro temos a representação do processo, assim como os botões que o controlam, tais como remover o processo do cenário, representado por um X, executar, parar e obter informações sobre este. Adicionalmente este poderá estar associado um ícone que o identifica, acima do seu nome, que no caso da Figura 26 é "River Simulatior". A esta componente estão ligados todos os seus parâmetros de entrada e de saída.

Figura 26 – Componentes do processo

No que diz respeito aos parâmetros de entrada e de saída, estes estão legendados com uma label que apresenta o nome do parâmetro seguido do tipo de dados esperado (e.g. double, integer, float, string, xml, image/png, etc.). As componentes apresentam no lado esquerdo um ícone que para alem de indicar se é um parâmetro de entrada ou e de saída, indica também se está presente algum valor. Este pode ser um valor que tenha sido introduzido pelo utilizador, um resultado de uma função de processamento ou um valor por defeito atribuído ao parâmetro. Quando o parâmetro está instanciado, o ícone passa a ser representado em verde como se pode ver na Figura 26. No lado direito é possível verificar a existência de outro botão que permite ao utilizador no caso de um parâmetro de entrada, verificar os dados que este contem (apresentado mecanismos para visualização de texto, imagens e outros tipos de dados tais como geometrias 2D e 3D que podem ser visualizadas no terreno) ou introduzi-los. A introdução pode incluir mecanismos desde o simples abrir de ficheiros do disco ao introduzir valores usando uma caixa de texto.

De modo a conectar parâmetros de entrada a parâmetros de saída ou outra fonte de dados basta que o utilizador, arraste a componente para cima de por exemplo, um parâmetro de saída ou uma selecção como ilustrado na Figura 25. No caso de uma ligação entrada-saida, e caso seja uma ligação compatível, verificada de um modo genérico através do tipo de dados esperado, ambas as componentes são substituídas por um novo elemento que deste modo simplifica a interface tal como se pode ver na transição da Figura 27 para a Figura 28. Este novo elemento é responsável por fazer a passagem de dados entre os dois ou mais processos.

Quando o utilizador por alguma razão pretende quebrar a ligação entre os dois processos, pode usar o botão assinalado na Figura 28. No caso em que o utilizador remove o processo, o sistema efectua a quebra de ligações automaticamente.

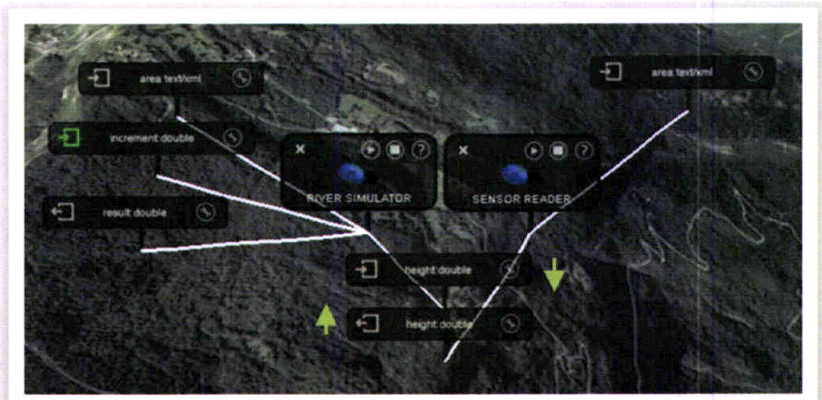

Figura 27 – Criar Ligações

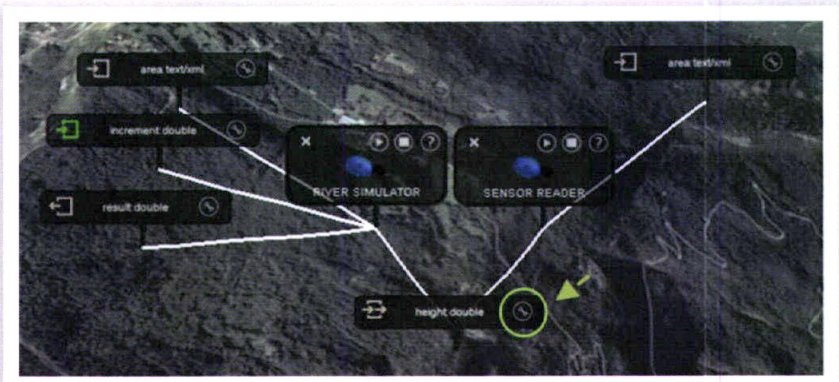

Figura 28 – Quebrar Ligações

Depois de construída a sequencia desejada de processos, o utilizador deve pedir a execução dos processos. Este procedimento pode ser feito executando o último processo, que automaticamente irá executar todos os que sejam necessários para a sua execução, ou o utilizador pode optar por executar um a um, começando do que não apresenta qualquer dependência, ou através de qualquer outra sequência. Quando um processo é executado, os dados do utilizador são convertidos mediante o necessário para que possam ser enviados através de XML, e usando o protocolo WPS ao servidor.

Enquanto os processos são executados é apresentado ao utilizador a informação de que estes estão a ser apresentados, através de uma barra de estado.

Detalhes técnicos

Quando os metadados sobre um processo são obtidos do servidor é instanciada uma classe que implementa a interface IProcess.

```
public abstract class IProcess {
        /**
         * Unambiguous identifier or name of a process, input, or output,
         * unique for this server.
         */
        public abstract String getIdentifier();

        /**
         * Title of a process, input, or output, normally available for display to a human
         */
```

```
public abstract String getTitle();

/**
 * Brief narrative description of a process, input, or output,
 * normally available for display to a human
 */
public abstract String getAbstract();

/**
 * Reference to more metadata about this process
 */
public abstract Metadata getMetadata();

/**
 * Release version of process (not of WPS specification)
 */
public abstract String getVersion();

/**
 * List of the required and optional inputs to this process
 */
public abstract ProcessInputs getInputs();

/**
 * List of the required and optional outputs from executing this process
 */
public abstract ProcessOutputs getOutputs();

/**
 * Indicates if all complex data output(s) from
 * this process can be stored by WPS server as web-accessible resources
 */
public abstract boolean isStoreSupported();

/**
 * Indicates if Execute operation response can be returned quickly with
 * status information
 */
public abstract boolean isStatusSupported();

/**
 * Initialization phase
 * @throws InitializationException
 */
```

```
    public abstract void initialize() throws InitializationException;

    /**
     * Process Activation Phase
     * @throws ActivationException
     */
    public abstract void activate() throws ActivationException;

    /**
     * Process Execution Phase
     * @throws ExecutionException
     */
    public abstract void execute() throws ExecutionException;

    /**
     * Dispose process
     */
    public abstract void dispose();

    /** For attaching information */
    public abstract void setProcessUnity(ProcessUnity unity);
    public abstract ProcessUnity getProcessUnity();
    public abstract void addDependency(IProcess process);
    public abstract void removeDependency(IProcess process);

}
```

Esta classe é responsável por definir os valores por defeito para os parâmetros, por remover todos os resultados associados a este processo assim como pela execução do processo. O procedimento de execução de um processo, é responsável pela obtenção dos resultados dos processos dos quais depende e que terá que executar se necessário, construção do pedido de execução a enviar ao serviço WPS e da interpretação dos resultados resultantes da resposta. No caso de um processo local em vez de elaborar o pedido a enviar, são executados os procedimentos ou algoritmos.

Os parâmetros de saída guardam uma referência aos dados obtidos do servidor OutputDescriptionType, assim como também uma lista com os parâmetros de entrada aos quais estão ligados (através da interacção do utilizador). A classe que representa um parâmetro de saída apresenta também uns métodos para verificar por exemplo se este parâmetro é compatível com qualquer um outro, contudo outras verificações principais, tais como se pertencem ao mesmo processo, são verificadas pelo controlador.

O excerto de código que se segue ilustra a estrutura de dados Output.

```
Public class Output implements TypeInfo {

        private RegistryTable<String, Object> table = new RegistryTable<String, Object>();
        private ArrayList<TypeInfo> outputs = new ArrayList<TypeInfo>();
        private String mimeType, encoding, schema, name;
        private OutputDescriptionType odt;

        public Output(String mimeType, String name) {
                this.mimeType = mimeType;
                this.name = name;
        }

        public void setEncoding(String encoding) {
                this.encoding = encoding;
        }

        public void setSchema(String schema) {
                this.schema = schema;
        }

        public void setOutputDescriptionType(OutputDescriptionType odt) {
                this.odt = odt;
        }

        public String getMimeType(){
                return mimeType;
        }

        public String getEncoding(){
                return encoding;
        }

        public String getSchema(){
                return schema;
        }

        public void setName(String name) {
                this.name = name;
        }

        @Override
        public String getName() {
                return name;
```

```
}

public boolean isComplexType(){
        return odt.isSetComplexOutput();
}

@Override
public Object getDefaultValue(){
        return getValue(name);
}

@Override
public Object getValue(String key) {
        return table.lookup(key);
}

@Override
public void putValue(String key, Object value){
        table.rebind(key, value);
        for(TypeInfo out : outputs){
                if(!key.equals(getName()))
                        out.putValue(key, value);
                else
                        out.putValue(out.getName(), value);
        }
}

public void addOutput(TypeInfo output) {
        if(!outputs.contains(output))
                outputs.add(output);
}

public void removeOutput(TypeInfo output) {
        outputs.remove(output);
}

@Override
public boolean isCompatible(TypeInfo p) {
        if(p instanceof Input){
                Input in = (Input)p;
                return in.getMimeType().equalsIgnoreCase(this.getMimeType());
        }
        return false;
}
```

```
        @Override
        public void reset(){
                table.reset();
        }

        @Override
        public String toString(){
                return "[OUTPUT TYPE=" +  mimeType + " Encoding=" + encoding + " Schema=" +
                                                schema + " name=" + name + "]";
        }

}
```

O Output pouco difere de um Input, sendo ambos actualizados a cada interacção do utilizador, se necessário.

Esta arquitectura, usa também sempre que possível os dado num modo nativo. Por exemplo, uma lista de características é sempre representada através de XML, contudo para efectuar operações sobre esta, é necessária a sua conversão de XML para um modelo de Objectos, o que consome algum tempo.

De modo a optimizar este problema foram criados 5 tipos nativos de dados:

```
public static final String NATIVE_GEOMETRY = "abstract_process_native_geometry";
public static final String NATIVE_DOUBLE_BUFFER = "abstract_process_native_dbuffer";
public static final String NATIVE_FLOAT_BUFFER = "abstract_process_native_fbuffer";
public static final String NATIVE_INT_BUFFER = "abstract_process_native_ibuffer";
public static final String NATIVE_SF = "abstract_process_native_sf";
```

E que sempre que disponíveis são passados juntamente com o valor no formato "oficial".

Assim sendo, se um processo necessita de aceder a um parâmetro de outro processo, inicialmente procura pelo formato nativo que lhe convêm usando a respectiva chave, se não existir pede o tipo por defeito. Por exemplo, no caso do XML recebe o documento XML com características, lê as características para o modelo de objectos e guarda-o com a chave NATIVE_SF no respectivo parâmetro, efectua as operações e passa o resultado ao próximo processo em formato nativo. O próximo processo procura pelo formato nativo se pretender efectuar alterações, caso contrário invoca o formato por defeito e um mecanismo interno efectua a conversão se necessário de um formato nativo que exista.

Esta optimização tem grande impacto, visto que é bastante comum a transformação XML -> características e Características -> XML (geralmente para enviar ao servidor).

No caso em que se esteja a trabalhar apenas com processos remotos, esta opção não representa grandes benefícios pois todos os processos tem por exemplo no fim que converter para XML, mas nos restantes casos implica aumentos significativos de performance como poderá ver nas secções seguintes.

3.9.3.1. Criação de novos processos on-fly

É também possível com as alterações sugeridas ao protocolo WPS, iniciar o modo de programação de novos algoritmos. Ao iniciar este modo, é adicionada uma widget no ecrã que possibilita trabalhar em 2D e em 3D ao mesmo tempo. Nesta widget inicialmente existe apenas uma componente que representa o algoritmo em si. Ao clicar na componente, é possível definir o algoritmo. Adicionalmente e se este necessita de parâmetros de entrada ou se produz algum parâmetro de saída, basta clicar no botão de adicionar (ver vídeo "Interface WPS" em anexo). Ao adicionar um parâmetro aparecerá uma nova componente ligada ao algoritmo. Quando se clica nesta componente, é possível definir o nome, tipo (parâmetro de entrada ou de saída), descrição, keywords e se necessário o valor por defeito. A descrição é importante para que o utilizador perceba para que o parâmetro serve, e as keywords são importantes para o sistema de pesquisa. O parâmetro terá uma representação diferente dependendo do seu tipo.

Se algum valor foi atribuído ao parâmetro, o ícone muda de cor para verde, como pode ser observado na componente à esquerda na Figura 29 (parâmetro de entrada).

Se alguma destas componentes for arrastada para fora da widget, então passa a fazer parte do contexto 3D e pode ser conectada a dados ou objectos neste. No caso da Figura 29, o parâmetro de entrada foi associado a uma área no terreno, dinamicamente seleccionada pelo utilizador.

Figura 29 – Construção de um processo

Quando o utilizador selecciona a componente que representa o algoritmo, surge uma caixa de texto (como ilustrado na Figura 30), onde para além de ser possível seleccionar a linguagem de programação, é possível obter um diagnostico em run-time através de uma compilação em memória dos erros de programação cometidos. Entre algumas das linguagens suportadas está o Java.

Adicionalmente, e de modo a que o programador apenas tenha que ter conhecimento do algoritmo, e não de como este tem que ser programado de modo a que funcione e encaixe no sistema modular de processos online, existem algumas macros que podem ser usadas: para passar o XML para objectos nativos - XML2FC(Object), objectos nativos para XML TO_XML(Object), para obter o valor de um

parâmetro de entrada através do seu respectivo nome GET_INPUT(String) ou para definir o valor de uma variável de saída SET_OUTPUT(String, String).

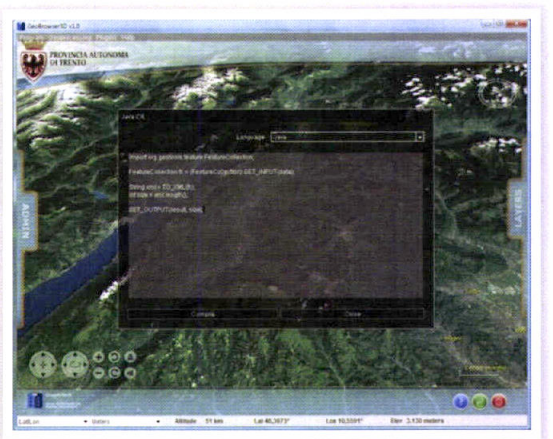

Figura 30 – Editor de código

Antes de enviar o código ao servidor, as macros se existentes são convertidas para código e este é compilado invocando a função compile(String, String, String). Se não existirem erros é então enviado para o servidor.

```
public Class<?> build(String code, String filename){
        JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
        DiagnosticCollector<JavaFileObject> diagnostics = new DiagnosticCollector<JavaFileObject>();
        StandardJavaFileManager fileManager = compiler.getStandardFileManager(diagnostics, null, null);

        Map<String, JavaFileObject> output = new HashMap<String, JavaFileObject>();
        ClassLoader loader = new MemoryClassLoader(output);
        JavaFileManager jfm = new MemoryFileManager(fileManager, output, loader);

        MemorySourceJavaFileObject src = new MemorySourceJavaFileObject(filename, code);
        JavaCompiler.CompilationTask task = compiler.getTask(null, jfm, diagnostics, null, null, Arrays.asList(src));
        boolean success = task.call();

        for (Diagnostic<? extends JavaFileObject> diagnostic : diagnostics.getDiagnostics()) {
                String msgError = diagnostic.getMessage(null);
                msgError = msgError.substring(msgError.lastIndexOf(":")+1);

                te.setError(true, msgError);
                te.getTextRendererData().setSelectionIndex((int) diagnostic.getStartPosition(),
                                (int) diagnostic.getEndPosition(), te.getAppearance());
```

```
            }

            try {
                    fileManager.close();
            }
            catch (IOException e) {
                    e.printStackTrace();
            }

            try {
                    return (!success) ? null : Class.forName(filename, false, loader);
            }
            catch (ClassNotFoundException e) {
                    return null;
            }
}

public void compile(String language, String code, String filename) {
            success = false;
            if (language.equals("Java")) {
                    if (build(code, filename) != null) {
                            controller.getWindowsManager().showMessage("Success", "Compiled with success!");
                            success = true;
                    }
            }
            else if(mgr.getEngineByName(language) != null){
                    ScriptEngine engine = mgr.getEngineByName(language);
                    try {
                            engine.compile(code);
                            success = true;
                    }
                    catch(ScriptException e){
                            controller.getWindowsManager().showMessage("Fail", e.getMessage());
                    }
                    if(success)
                            controller.getWindowsManager().showMessage("Success", "Compiled with success!");
            }
            else if (compileLanguageCode(language , code, filename) != null){
                    controller.getWindowsManager().showMessage("Success", "Compiled with success!");
                    success = true;
            }
}
```

Depois de o processo ser enviado ao servidor usando a especificação apresentada na secção anterior, fica automaticamente publicado online (se não acontecer nenhum problema) e pode então ser usado a partir do menu no cliente que é automaticamente populado.

Ao seleccionar o recém-adicionado processo, este é adicionado ao ambiente 3D, mantendo a informação com a qual foi criado, tal como, a selecção de determinada área associada a um parâmetro de entrada, como ilustrado na Figura 31.

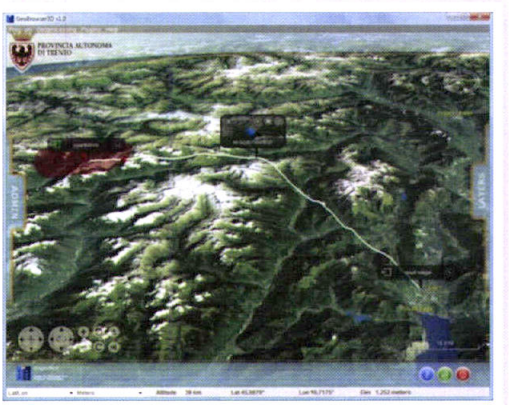

Figura 31 – Execução de um processo criado

Depois diversos processos podem ser interligados, conectando parâmetros de entrada com parâmetros de saída ou com dados provenientes do ambiente 3D. No caso ilustrado na imagem que se segue, foi conectado ao recém-criado algoritmo, um algoritmo que cria uma inundação quando especificada a elevação das águas. O resultado do algoritmo criado na Figura 30, que apenas gera um número quase random, poderia ser um algoritmo complexo, que inclusive poderia aceder a serviços meteorológicos e processar esses dados de modo a prever ou calcular a actual ou futura altura das águas.

Figura 32 – Concatenação do processo criado com outros processos

Capítulo IV: Caso de Estudo

Nesta secção é apresentado um caso de estudo do uso prático, adoptado para validar a eficácia e ilustrar o potencial da interface.

Neste estudo foi pedido aos utilizadores que simulassem uma inundação através do aumento da altura real do rio Adige em um determinado valor (por exemplo 25 mts) de modo a identificar as zonas de perigo e encontrar as populações afectadas pela potencial inundação.

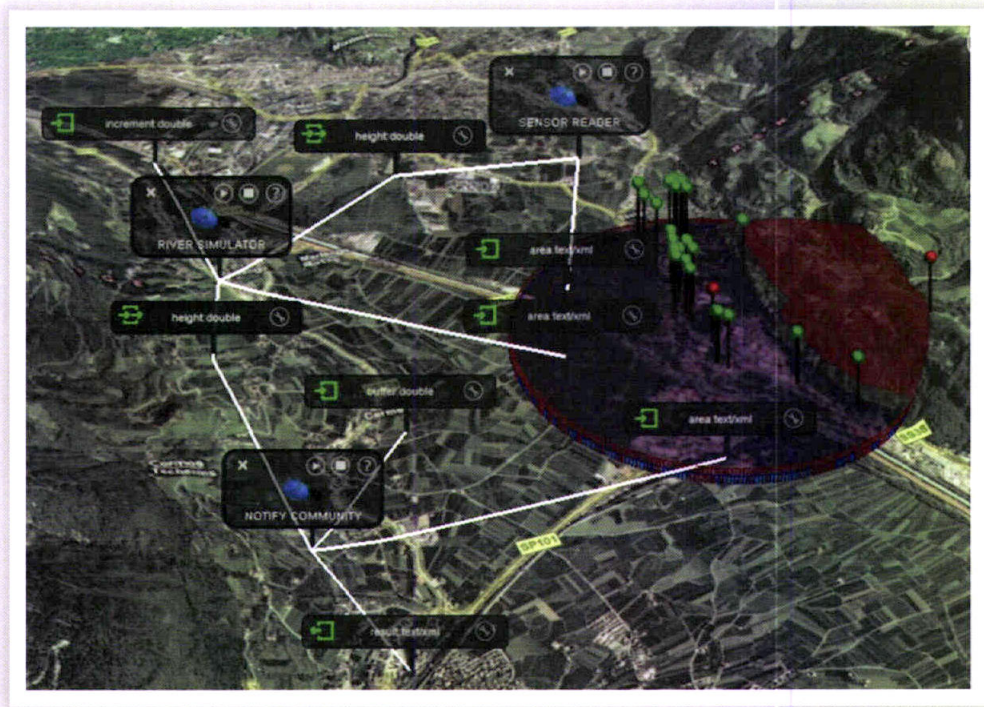

Figura 33 – Simulação de uma Inundação

Uma das possíveis soluções é ilustrada na Figura 33. Nesta combinação de processos, o primeiro processo (a começar da parte superior) opera como um filtro que identifica um sensor disponível dentro de uma região seleccionada pelo utilizador e respectivamente conectada ao processo. Os dados relevantes do sensor/processo [dados de saída da 1ª componente], relativos à elevação das águas, são passados a uma segunda unidade de processo que simula a inundação de modo visual. O processo de simulação usará a área seleccionada (que pode não ser a mesma) para delimitar os resultados da inundação. Os dados de saída do processo, que representam a elevação máxima da inundação simulada, são passados com dados de entrada a um último processo, que juntamente com uma selecção de uma área vai verificar na base de dados espacial a lista de ruas na área afectada, invocando depois um serviço web que efectua uma procura nas páginas brancas usando a rua de modo a identificar o número de telefone das pessoas em perigo. Como esta informação é real (nome, sobrenome, endereço e

número de telefone), por motivos de privacidade essa informação não é aqui ilustrada. O endereço exacto da pessoa, que inclui informações como o número de porta, etc., é passado a um serviço web de geocodificação da Google, que devolve as coordenadas geográficas, que serão usadas para obter a elevação da pessoa através de dados LIDAR. As pessoas dentro da área inundada serão assinaladas com um pin verde, que devolve as informações sobre a pessoa ao ser seleccionar com o rato.

Depois de todos os processos estarem conectados, o utilizador necessita apenas de executar o último processo da cadeia, que recursivamente ira executar todos os processos que representam uma dependência, ou executar individualmente cada processo.

Quando o processamento termina, é apresentada uma janela que lista os nomes e contactos de todas as pessoas que potencialmente estão em perigo. Posteriormente o operador pode criar planos de evacuação, tendo em consideração a distribuição das pessoas, estradas em trabalhos, caminhos óptimos, etc., que podem ser calculados de maneira similar recorrendo a muita a informação que pode ser encontrada online.

4.1. Teste e Validação

O teste de usabilidade baseado no caso de uso prévio envolveu 10 utilizadores, de três países diferentes (Portugal, Itália e Irão), com especialidades diferentes e onde nenhum tinha qualquer experiência em sistemas SIG. 10% da amostra tinham idades inferiores aos 25 anos, 60% entre 25 e 35, 10% entre 35 e 45 e 10% eram mais velhos que 45. Sete entre as dez pessoas envolvidas no teste eram masculinas. 60% eram estudantes, 30% pós-graduados e os restantes 10% apenas tinha o secundário.

De modo a evitar troca de influências os utilizadores foram convidados individualmente para a sala de testes e introduzidos ao sistema e funcionalidades durante menos de 1 minuto. Depois foram pedidos a efectuar a tarefa definida com o caso de estudo numa área localizada nas proximidades da cidade de Merano em Itália. O teste tinha como duração máxima 20 minutos.

Depois da sessão de teste, ocorreu uma sessão de discussão sobre o sistema, seguida do preenchimento de um questionário previamente preparado. Todas as sessões e discussões foram gravadas.

4.2. Análise dos resultados

Esta secção ilustra os resultados de um teste ISOMÉTRICO (ISOMETRICS, 2006), baseado nos padrões ISO. O teste é dividido em sete categorias onde o utilizador pode decidir não dar opinião sobre a sentença, contudo em todos os casos os utilizadores deram uma opinião.

4.2.1. Adequação para a tarefa

Esta secção mostra que a interface pode em geral ser considerada satisfatória para a tarefa. Contudo alguns problemas menores ainda têm que ser resolvidos para que o software seja considerado completamente satisfatório.

Esta posição foi expressada de um modo claro pelos utilizadores como é visível na Figura 34 onde pontos como A.3 - As funções implementadas no software suportam me na execução do meu trabalho – tiveram uma ausência completa de "discordância" (vermelho) ou "neutro" (em verde).

É também possível notar a ausência completa de "acordo" (azul) ou "neutro" por (em verde) na sentença A.1 - O software força-me a executar tarefas que não são relacionadas a meu trabalho actual.

Também é interessante o resultado de A.5 – É necessário executar muitos passos diferentes para lidar com uma determinada tarefa - que mostra uma discordância clara (1.90) indicando a eficiência da interface desenvolvida. Além disso muitos utilizadores consideraram que os comandos são fáceis de encontrar (A.12 - 4.50), que o software é bem adequado às exigências do trabalho (A.7 - 4.20) e que mais trabalho deveria ser feito no que diz respeito à informação exibida sobre a tela (A.8 - 3.80).

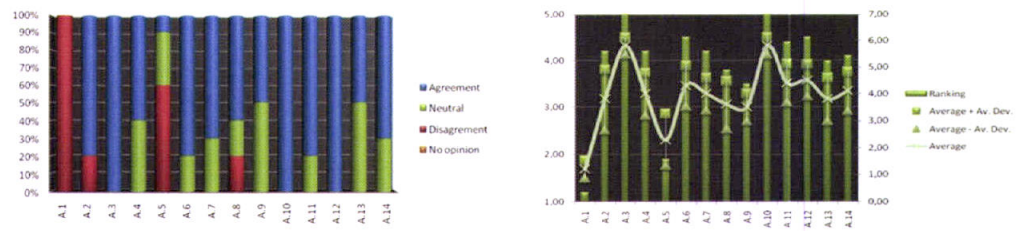

Figura 34 – Adequação para a tarefa

4.2.2. Autodescritividade

Em termos de autodescritividade o teste mostrou que os utilizadores estão satisfeitos com a interface. Porém ficou claro que é necessário um esforço futuro de modo a melhorar o sistema na possibilidade de obter informações mais facilmente sobre um certo campo de entrada (B.3 - 3.20).

Felizmente a interface provou ser bastante clara de usar com base na informação mostrada na tela (B.9 - 4.10) com o utilizador a perceber claramente o conceito transmitido pela interface (B.10 - 4.00).

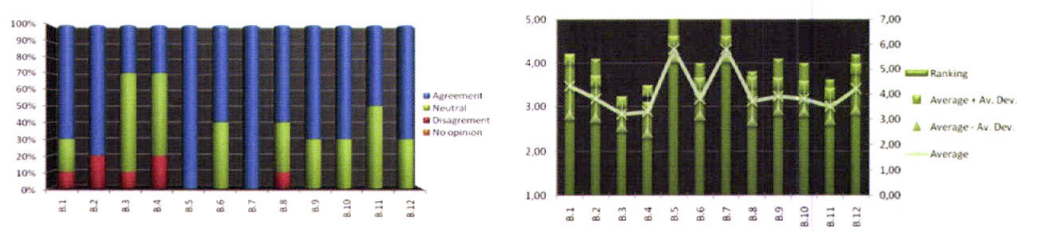

Figura 35 – Autodescritividade

4.2.3. Controlabilidade

No que diz respeito à controlabilidade, a interface provou que no seu uso é fácil mudar entre funções diferentes do menu (C.2 - 4.10) e voltar ao estado principal (C.3 - 4.30).

Também a facilidade de navegação recebeu um bom resultado, que é considerada essencial para o uso do sistema (C.1 - 4.50).

É possível notar como seria possível melhorar a controlabilidade do sistema através da possibilidade de uso de teclas de atalho (C.10 - 2.70).

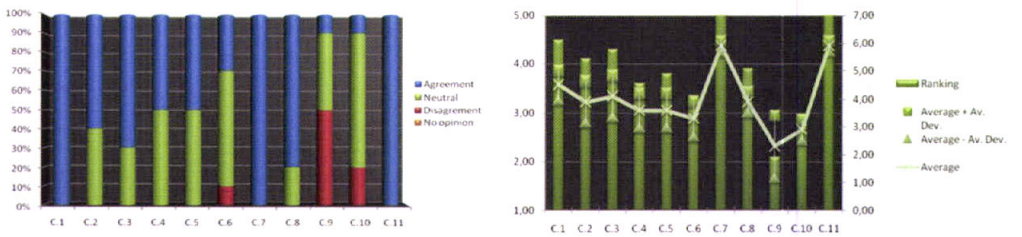

Figura 36 – Controlabilidade

4.2.4. Conformidade com as expectativas do utilizador

Os utilizadores indicaram claramente que o sistema não é inconsistente e devido a isso não é difícil de usar (D.1 - 2.90). Este ponto parece ser bastante crucial visto que os utilizadores indicaram também a importância de consistência (D.4 - 3.10) com outros softwares com que eles estão familiarizados.

Contudo em geral existe uma grande ambiguidade entre as desta secção. Na realidade várias questões registaram um número muito alto de avaliações neutras (entre 30%-70%). Porém a maioria restante concorda com a possibilidade para se antecipar qual a operação que aparecerá em uma sucessão de processos (D.2 - 4.20) e na previsibilidade dos resultados (D.6 - 4.00).

Quase uma marca negativa é dada ao predizer quanto tempo o software precisará para efectuar uma operação (D.3 - 3.10).

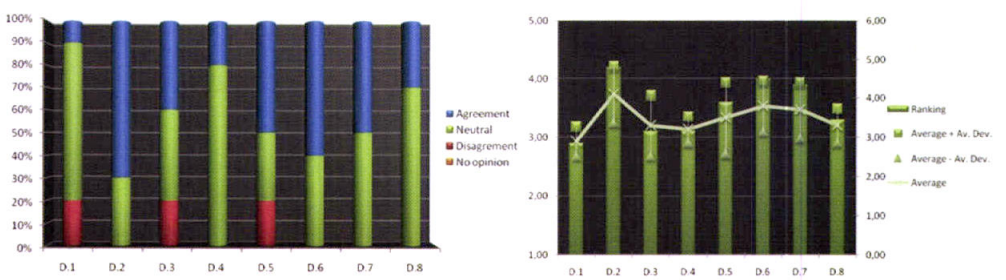

Figura 37 – Conformidade com as expectativas do utilizador

4.2.5. Tolerância a erros

Nos resultados do teste está claro que é necessário um esforço adicional para suportar melhor a tolerância de erros. Na realidade os utilizadores assinalaram que enganos pequenos às vezes têm consequências sérias (3.00).

Além disso e mais importante, há uma sensação de que quando o utilizador tenta executar uma operação destrutiva este não é forçado a confirmar a acção. É também importante sublinhar o resultado que é particularmente pobre (2.25).

Uma marca positiva é dada pelos utilizadores sobre o facto de que durante o uso do sistema nunca ocorreu um erro de software (por exemplo um bloqueio do sistema / programa ou um estado de diálogo indefinido) (4.30) demonstrando, a importância para o utilizador de ter um sistema estável e seguro. Deste ponto de vista podemos declarar que a estabilidade de sistema atingiu o seu nível mais elevado.

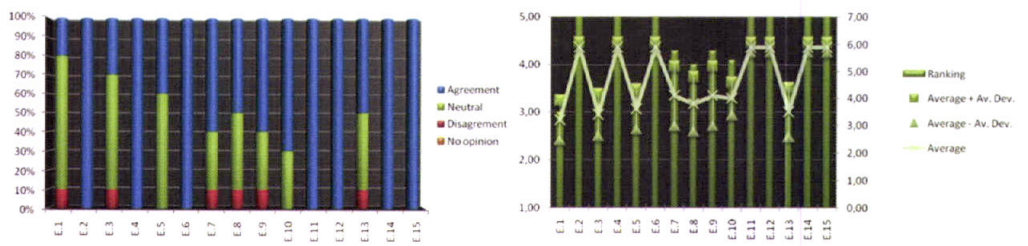

Figura 38 – Tolerância a erros

4.2.6. Adequação para a personalização

Devido ao facto de a interface não suportar individualização o teste mostrou que os utilizadores não estão satisfeitos com a possibilidade de personalização oferecida para os menus e outros widgets GUI-relacionados (ranking de 2.6).

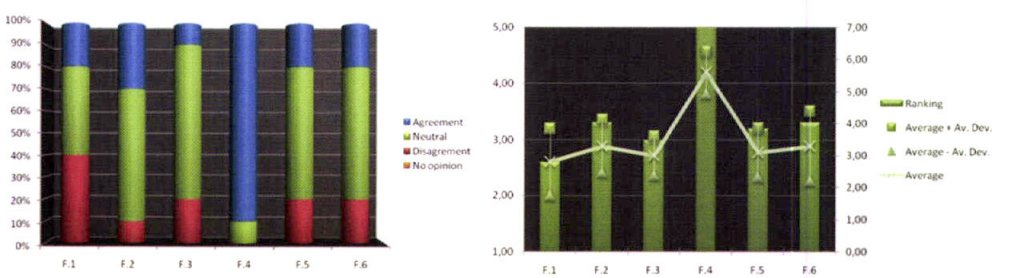

Figura 39 – Adequação para personalização

4.2.7. Adequação para a aprendizagem

Semelhantemente para a secção anterior, também aqui é possível notar que um grande número de utilizadores que expressão uma opinião neutra sobre assunto. Este evento leva nos à consideração que esta secção, semelhantemente à anterior, é de pouca importância para os utilizadores.

Os utilizadores restantes indicaram porém que eles não precisaram de muito tempo para usar o software (2.50), que o software é supostamente fácil de reaprender e usar depois de uma interrupção prolongada (4.40) e que não há muita necessidade de se lembrar grandes detalhes para usar o software correctamente (2.50).

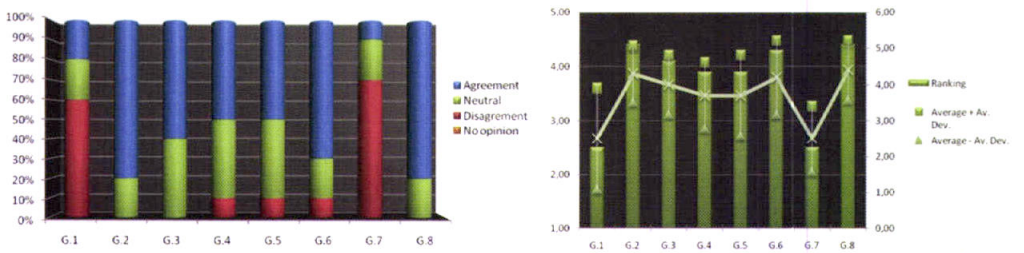

Figura 40 – Adequação para a aprendizagem

Capítulo V: Conclusões e Perspectivas Futuras

Hoje interoperabilidade está a começar a tornar-se uma realidade graças aos vários esforços de harmonização internacionais, que tentam resolver o problema oferecendo padrões de computação abertos para administração e processamento de informação geográfica. O trabalho aqui apresentado mostra como interoperabilidade é fundamental para ligar muitos sistemas de grandes quantidades de conjuntos de dados complexos, assim como é fundamental a criação de novos mecanismos de processamento e visualização de dados.

Numa tentativa de melhorar o problema foi apresentada uma infra-estrutura completa para aceder, visualizar e processar informação geográfica dentro de um 3D, onde foi possível observar que operar directamente dentro de um contexto de georreferenciado, permite ao utilizador de um modo natural, verificar em tempo real dados e resultados, assim como explorar os diferentes cenários / possíveis alternativas por um verdadeiro processo analítico.

A aplicação desenvolvida permite também que múltiplas entidades possam em tempo real e de um modo colaborativo editar os dados, criar novos algoritmos ou juntarem os seus recursos já existentes (servidores com algoritmos específicos para a sua actividade) de modo a usa-los em sistema único, que pode ser executado da internet e que não supera os 5Mb de tamanho. Passa assim a ser possível desenvolver actividades que anteriormente eram efectuadas através de uma colaboração que não era explorada ao máximo, e que era caracterizada pelo seu aspecto caótico.

Por fim, a experiência mostrou que os utilizadores indicaram claramente que a interface pode ser em geral considerada satisfatória e autodescritiva para uma determinada tarefa, quebrando a complexidade da maioria dos sistemas SIG standards e permitindo que qualquer pessoa, mesmo não sendo especialista em sistemas SIG, o possa usar mesmo com pouco treino.

Em geral foram atingidos todos os objectivos propostos para esta tese, contudo com a realização desta tese abriram-se novas possibilidades que permitem melhorar ainda mais o sistema. Uma que é bastante relevante é a possibilidade de programar o código dos algoritmos também de modo visual como por exemplo se pode observar na aplicação Alice 2.0 (ver vídeo AliceScreencast.mov que vem com o suporte digital). Através desta abordagem um utilizador não tem qualquer necessidade de se lembrar da sintaxe da linguagem de programação, tornando assim a aplicação ainda acessível aos utilizadores.

Também os protocolos OGC oferecem pouco suporte para questões como de internacionalização. Em melhorias futuras estes devem possibilitar aos clientes a escolha do idioma, sendo por exemplo possível obter o nome das camadas em diferentes línguas, suportadas pelo servidor.

Anexo I. Serviços Web OGC

Quando um processo de uma aplicação é executado num computador diferente, localizado na rede, então está inserido num contexto de computação distribuída. O processo usa a rede (i.e. as interfaces, protocolos, esquemas etc.) para a transmissão de dados. Estes processos de computação online são chamados de Serviços Web. A especificação de Serviços Web *OGC* é a definição das interfaces e protocolos padrão, codificações e esquemas, para Serviços Web que controlam informação georreferenciada (GEO World, 2003).

OGC Web Services suporta actualmente uma plataforma de computação distribuída, a Internet e o Protocolo de Transferência de Hipertexto (HTTP). Como a área de geomática é uma área grande, uma especificação de interface não é suficiente suportar livremente uma troca de informação espacial. Por isso, existem várias especificações de implementação OGC, com finalidades distintas.

1. Regras para a formulação de um pedido

No que diz respeito a comunicação por HTTP é importante começar por referir que o URL deve estar em conformidade com a descrição em IETF RFC 2616 e que o servidor deve suportar pelo menos o método GET sendo o método POST opcional no caso do serviço WMS.

Alguns caracteres tais como "&", "=", "," e "+" são de uso reservado na especificação do URL IETF RFC 2396. Caso sejam usados devem ser codificados de acordo com as regras definidas em IETF RFC 2396 de modo a que não entrem em conflito com a sua funcionalidade na especificação. O servidor deve ser capaz de descodificar quando necessário este caracteres.

O URL ou tem que terminar com um "?"' (na ausência de parâmetros) ou um "&", e apresenta um formato semelhante a http://endereço[:porto]/caminho?{parametro[=valor]&}, onde [] denota 0 ou 1 ocorrência de uma parte opcional e {} denota 0 ou mais ocorrências.

A resposta de um servidor é sempre um ficheiro que é transmitido pela internet para o cliente. Este ficheiro pode conter texto ou o ficheiro que representa a imagem do mapa.

O nome de um parâmetro não necessita de respeitar maiúsculas/minúsculas, contudo o seu valor deve-o fazer. Neste documento os nomes dos parâmetros são apresentados tipicamente em maiúscula para claridade tipográfica, não como uma exigência.

Os parâmetros em um pedido podem ser especificados em qualquer ordem. Quando um parâmetro está duplicado ou tem valores contraditórios, a resposta do servidor pode ser indefinida. Um servidor deve também estar preparado para encontrar parâmetros adicionais no pedido, que não fazem parte do padrão.

Para enviar um pedido a um servidor, a única coisa necessária é o acesso ao Hypertext Transfer Protocol (HTTP), especificando o pedido na forma de um URL. O servidor deverá interpretar o pedido e responder (Open Geospatial Consortium, 2008). Como o URL deve ser construído depende do pedido escolhido. Comum a todos os URL é a existência de um número de versão e um parâmetro que especifica ao servidor o tipo de pedido.

2. HTTPS

Em vez de oferecer serviços web através do protocolo de HTTP, um servidor pode também oferecer serviços web que usam HTTPS. HTTPS é o HTTP sobre um canal de comunicação seguro que permite transferir informação codificada entre máquinas pela World Wide Web.

O uso de HTTPS não afecta a descrição dos pedidos e respostas descrita nas especificações mas pode exigir acções adicionais no cliente e no serviço para iniciar a comunicação segura.

3. Parâmetros comuns

Esta interface especifica parâmetros que são comuns aos diversos serviços web do OGC.

3.1. SERVICE

O parâmetro obrigatório SERVICE indica qual o tipo de serviço a usar (pode tomar valores como "WMS", "WFS", "WPS", etc.) e que está disponível num servidor particular.

3.2. REQUEST

O parâmetro obrigatório REQUEST indica qual operação do serviço que está a ser invocada. O valor será o nome de uma das operações oferecidas pelo servidor.

3.3. VERSION

O parâmetro VERSION especifica a versão do protocolo a usar. O número de versão é composto por três números não negativos separados por pontos decimais na forma "x.y.z". Os números "y" e "z" não devem exceder 99. Este número de versão é alterado a cada revisão do padrão, e deve ser usado para negociar uma versão entre o cliente e o servidor em que ambos possam comunicar.

O número de versão deve aparecer em pelo menos dois lugares: nos metadados do serviço e na lista de parâmetros no pedido do cliente ao servidor. O número de versão usado no pedido de um cliente a um servidor deverá ser igual a um dos números de versão suportados por este. Um servidor pode suportar várias versões que podem ser negociadas com os clientes de acordo com as regras de negociação:

1. Se o servidor implementar o número de versão pedido, o servidor tem que enviar aquela versão.
2. Se o pedido do cliente é para uma versão desconhecida maior que a versão mais baixa que o servidor entende, o servidor tem que enviar a versão mais alta menor que a versão pedida.
3. Se o pedido do cliente é para uma versão abaixo de qualquer uma conhecida pelo servidor, o servidor tem que enviar a mais baixa versão que suporta.

4. Se o cliente não reconhecer o número de versão novo enviado pelo servidor, pode deixar de comunicar com o servidor ou pode enviar um novo pedido com um número de versão novo que o cliente entende, mas que é menor que o enviado pelo servidor (se o servidor tivesse respondido com uma versão inferior).
5. Se o servidor tivesse respondido com uma versão mais alta (porque o pedido era para uma versão abaixo de que qualquer uma conhecida pelo servidor), e o cliente não entende a versão mais alta proposta, então o cliente pode enviar um novo pedido com um número de versão mais alto que o enviado pelo servidor.

O processo é repetido até que uma versão mutuamente compreendida é alcançada, ou até que o cliente determine que não vai ou não pode comunicar com aquele servidor de particular.

3.4. ACCEPTVERSIONS

Parâmetro opcional que enumera todas as versões que são aceites pelo cliente, por ordem de preferência.

3.5. FORMAT

O parâmetro opcional FORMAT declara o formato desejado para uma resposta a uma operação. Os valores suportados são listados em um ou mais elementos <Request><GetCapabilities><Format> nos metadados do serviço.

Se o pedido especificar um formato não suportado pelo servidor, o servidor deverá responder com o formato por defeito *text/xml*.

3.6. ACCEPTFORMATS

Parâmetro opcional que enumera todos os formatos que são aceites pelo cliente, por ordem de preferência

3.7. EXCEPTIONS

O parâmetro opcional EXCEPTIONS declara o formato no qual são devolvidos os erros. Se este parâmetro estiver ausente no pedido, o valor por defeito é "XML".

Um serviço deverá suportar um ou mais formatos para a apresentação de excepções declarados numa lista de elementos <Format> dentro do elemento <Exceptions> nos metadados do serviço. O primeiro destes formatos deve ser oferecido por todos os serviços; os formatos de excepção restantes são opcionais. Um servidor pode emitir uma excepção de serviço no formato definido por defeito (XML) se um pedido especificar um formato de excepção não suportado pelo servidor.

XML (Obrigatório)

A excepção é enviada ao cliente no formato XML, como especificado no Anexo VI. Este é o formato de excepção por defeito se nenhum é especificado no pedido. Os códigos de excepção dependem do tipo de serviço.

INIMAGE (Opcional)

Se o parâmetro EXCEPTIONS tem o valor INIMAGE, o serviço ao descobrir um erro deve devolver um objecto do MIME type especificado no parâmetro de FORMAT cujo conteúdo descreve a natureza do erro. No caso de um formato de imagem, a mensagem de erro pode aparecer escrita ou representada com uma imagem ilustrativa na imagem devolvida.

BLANK (Opcional)

Se o parâmetro EXCEPTIONS tem o valor BLANK, o serviço ao descobrir um erro deve devolver um objecto do tipo especificado em FORMAT sem qualquer conteúdo. No caso de um formato de imagem, a resposta será uma imagem que contém pixéis de uma só cor (a cor de fundo). No caso de um formato de imagem com transparência, todos os pixéis serão transparentes. No caso de um formato de um elemento gráfico, nenhum elemento gráfico será incluído na resposta.

3.8. UPDATESEQUENCE

O parâmetro opcional UPDATESEQUENCE tem como objectivo manter a consistência da cache. O valor pode ser um inteiro ou uma string, que por exemplo representa um timestamp no formato ISO 8601:2004 (Exemplo: 2000-06-23T20:07:48.11Z). O servidor pode incluir um valor de UpdateSequence nos metadados do serviço. Se presente, este valor dever ser aumentado quando são efectuadas mudanças às capacidades do serviço (por exemplo, quando são acrescentados novos mapas). O cliente pode também incluir este parâmetro em seu pedido de GetCapabilities.

4. Pedido GetCapabilities

Quando invocado num serviço, a resposta para o pedido de GetCapabilities será um documento de XML que contém os metadados sobre o serviço, formatados de acordo com o esquema de XML. O esquema especifica o conteúdo obrigatório e opcional nos metadados sobre o serviço e como o conteúdo é formatado. O documento de resposta de capacidades contém pelo menos as seguintes secções:

4.1. Identificação do Serviço (ServiceIdentification)

A secção de identificação do serviço fornece informação sobre o próprio serviço, como definido na especificação OWS (Open Geospatial Consortium, 2009).

Esta informação inclui título do serviço, resumo, palavras-chave, tipo de serviço, versões suportadas, taxas, restrições de acesso, etc.

4.2. Secção Service Provider

A secção de fornecedor de serviço fornece metadados sobre a organização responsável pelo serviço como definido na especificação OWS (Open Geospatial Consortium, 2009).

Esta secção inclui informação como o nome do responsável, website e contactos.

4.3. Secção Operation Metadata

A secção de metadados das operações fornece metadados sobre as operações definidas nesta especificação e implementadas pelo servidor. O conteúdo da mesma é definido também segundo a especificação OWS (Open Geospatial Consortium, 2009). Estes metadados incluem o DCP, parâmetros e restrições para cada operação.

Anexo II. Web Map Service (WMS)

Um Web Map Service (WMS) pode ser descrito como: "um exemplo cujo comportamento segue uma especificação de interface, que descreve os pedidos e respostas a serem produzidos por um servidor de mapas standard" (ESRI, 2003).

Por outras palavras, é um serviço online que tem como finalidade a produção de mapas sobre dados espacialmente referenciados, dinamicamente da sua informação geográfica.

Este padrão define um "mapa" como sendo um retrato da informação geográfica em um arquivo de imagem digital, que pode ser facilmente exibido em qualquer monitor de um computador. Um mapa não é definido como sendo dados georreferenciados por si próprio; é uma representação visual dos dados geográficos. Os mapas produzidos são geralmente apresentados em um formato pictórico como PNG, GIF ou JPEG, ou ocasionalmente como elementos vectoriais como Scalable Vector Graphics (SVG) ou Web Computer Graphics Metafile (WebCGM) (Open Geospatial Consortium, 2008).

Apesar de este serviço poder fornecer alguns dados sobre as características presentes em um mapa, o serviço não tem como objectivo servir características ou os valores dos dados.

Este padrão define três operações, uma que devolve informações sobre as operações e informação geográfica disponível no servidor; outra que devolve um mapa onde os parâmetros geográficos e dimensões estão bem definidos (Fallman, 2004); e uma terceira, que é opcional e que devolve informações sobre características em particular apresentadas no mapa (Open Geospatial Consortium, 2008).

Estas três operações podem ser invocadas usando um Web browser e efectuando pedidos na forma de URL. O conteúdo do URL depende da operação requisitada. Por exemplo, num caso em que se pretende obter um mapa, o URL deve indicar que a operação é a de obter um mapa e os parâmetros específicos a esta operação tais como a informação que deve estar contida no mapa, a área da Terra que deve ser apresentada, o Sistema referencial e dados relativos à imagem devolvida como resultado como a altura e largura. Quando dois ou mais mapas são produzidos com os mesmos parâmetros geográficos e com as mesmas dimensões de output, os resultados podem ser sobrepostos com exactidão e criar um mapa composto. O uso de formatos de imagem que suportam fundos transparentes (exemplo: PNG ou GIF) permite que mapas que estão abaixo sejam visíveis. Para além disso, mapas individuais podem ser requisitados de diferentes servidores.

A Especificação de Implementação WMS é uma grande vantagem para todos os clientes de mapas web distribuídos pelo mundo, mas também oferece vantagens aos fornecedores de dados espaciais. Como os clientes de WMS podem escolher livremente e podem aceder a servidores de WMS por uma interface comum, o mesmos dados espaciais não tem que ser servidos de um servidor de WMS concreto. Este facto ganha tempo e custos aos proprietários de servidores de WMS e provedores de dados espaciais (Fallman, 2004).

1. Sistema Referencial por Coordenadas (CRS)

O serviço WMS deve suportar pelo menos um CRS, e mapas de múltiplos servidores só devem ser sobrepostos se todos os servidores seleccionados usarem o mesmo CRS. O Padrão não define um CRS específico como obrigatório, ficando a escolha deste ao critério do que for mais útil ou der mais jeito dentro da sua finalidade específica.

De modo a maximizar a interoperabilidade entre os servidores estes devem também suportar coordenadas geográficas através de sistemas de coordenadas geocêntricas como "CRS:84" (que corresponde ao WGS 84. A longitude e latitude geográficas são expressas em ângulos, com a longitude a variar entre –180º até +180º e latitude entre –90º até +90º), "EPSG:4326" (que corresponde ao WGS 84. Latitude e depois longitude) ou outros sistemas baseados no ITRF.

Os namespaces validos para o parâmetro CRS são:

CRS (CRS, 2006)

O identificador de CRS "CRS" é constituído pelo namespace "CRS" seguido do símbolo ":" e um código numérico. Estas definições estão na forma especificada por ISO 19111.

Exemplo: "CRS=CRS:84" que corresponde ao WGS 84.

EPSG (EPSG, 2008)

O identificador de CRS "EPSG" recorre ao CDG geodésico do European Petroleum Survey Group (EPSG) que define identificadores numéricos para muitos dos Sistemas Referenciais de Coordenadas comuns. O identificador é composto pelo namespace "EPSG" seguido do símbolo ":" e um código numérico.

Exemplo: "CRS=EPSG:4326" refere a WGS 84 latitude seguida de longitude geográfica. Quer dizer, neste CRS o eixo de x corresponde a latitude e o eixo de y a longitude. Este será também o CRS usado no protótipo 3D, pois coordenadas esféricas são mais fáceis de representar numa ambiente 3D.

AUTO2

O identificador CRS "AUTO2" é usado para sistemas referenciais de coordenadas "automáticos"; isto é, um CRS onde o utilizador pode definir o centro da projecção.

O identificador CRS "AUTO2" na sua forma completa é composto pelo prefixo "AUTO2:", um identificador numérico do namespace AUTO2, um número indicando qual o factor a aplicar para converter entre as unidades da bounding box e as unidades CRS AUTO2, e os valores para o centro da projecção (longitude e latitude em graus).

AUTO2:auto_crs_id,factor,longitude,latitude

O factor de conversão deve de ser positivo e não nulo (maior que zero). Se for igual a 1, então as unidades da BBOX são as mesmas que as usadas na definição CRS, caso contrário o factor é a razão entre as unidades BBOX e as unidades CRS.

Num pedido GetMap é especificado o CRS AUTO2 na sua forma completa. Contudo nos metadados do serviço é apenas especificado o identificador numérico do namespace AUTO2.

Exemplo: Um servidor indica que suporta CRS AUTO2 incluindo elementos como "<CRS>AUTO2:42003</CRS>" nos metadados do serviço. Um cliente pode emitir um pedido de GetMap para um mapa neste CRS, com a Bounding Box em metros, centrada a -100 graus de longitude e 45 graus de latitude, usando o parâmetro "CRS=AUTO2:42003,1,-100,45". No caso de uma conversão para U.S. feet o factor deverá ser 0.3048006096012192 em vez de 1.

"CRS=AUTO2:42003,0.3048006096012192,-100,45"

Informação Geográfica sem um CRS definido

Um servidor pode oferecer informação geográfica bidimensional cuja referência espacial é indefinida ou não é precisa. Por exemplo, um mapa histórico feito à mão pode representar uma área da Terra, mas pode não empregar um sistema de coordenadas de referência moderno, e uma fotografia aérea pode não ser georreferenciada com precisão. Estes tipos de informação devem ser tratados como uma imagem de CRS igual a "CRS:1". Os clientes não devem tentar combinar uma camada para qual CRS=CRS:1 com outra camada.

2. Bounding Box (Região Delimitadora)

Quando se quer obter alguma coisa sobre um mapa deve também ser especificada a região delimitadora (bounding box), que é constituída por dois pares de coordenadas no CRS especificado para a camada. O primeiro par de valores especifica os valores mínimos para as coordenadas, e o segundo par os valores máximos.

Em teoria existem mais métodos para especificar uma região do mapa sem ser através do uso de dois pontos de cantos, tais como por exemplo um ponto central e um raio. Contudo no pedido a um servidor WMS que implemente o Padrão, estes dados devem ser convertidos para uma Bounding Box.

Os valores de uma bounding box aparecem nas seguintes entidades

- No elemento <BoundingBox> nos metadados do serviço
- No parâmetro BBOX no pedido GetMap
- No parâmetro BBOX na parte do pedido de mapa no pedido de GetFeatureInfo

É também importante referir que a Bounding Box deve ter uma área superior a 0.

Um exemplo de uma Bounding Box para uma Camada (Layer) que representa a Terra toda no CRS=CRS:84 pode ser escrita como:

<BoundingBox CRS="CRS:84" minx="-180" miny="-90" maxx="180" maxy="90">

E o respectivo parâmetro BBOX pode ser representado com BBOX=-180,-90,180,90.

Caso por exemplo o CRS seja EPSG:4326, este pode ser representado por:

<BoundingBox CRS="EPSG:4326" minx="-90" miny="-180" maxx="90" maxy="180">

E o parâmetro BBOX por BBOX=-90,-180,90,180.

3. Camada (Layer)

Uma <Layer> pode ter zero ou mais dos seguintes atributos de XML: *queryable*, *cascaded*, *opaque*, *noSubsets*, *fixedWidth*, *fixedHeight*. Todos estes atributos são opcionais e tem como valor por defeito 0. Cada um destes atributos pode ser herdado ou ser substituído em sub-camadas. O significado de cada atributo é resumido na Tabela 2.

Atributo	Valores permitidos	Significado (0 é o valor por defeito)
queryable	0, false, 1, true	0, false: a camada não é interactiva. 1, true: a camada é interactiva.
cascaded	0, inteiro positivo	0: a camada não está a ser retransmitida em cascata. n: a camada está a ser retransmitida *n* vezes em cascata.
opaque	0, false, 1, true	0, false: os dados representam características vectoriais que provavelmente não preenchem todo o espaço do mapa. 1, true: o mapa é significantemente ou completamente opaco.
noSubsets	0, false, 1, true	0, false: WMS pode também reproduzir subpartes do mapa. 1, true: WMS apenas pode reproduzir um mapa como uma única área
fixedWidth	0, inteiro positivo	0: O servidor consegue produzir mapas com comprimentos arbitrários. Inteiro positivo: O servidor apenas produz mapas para o comprimento estipulado.
fixedHeight	0, inteiro positivo	0: O servidor consegue produzir mapas com uma altura arbitrária. Inteiro positivo: O servidor apenas produz mapas para a altura estipulada.

Tabela 2 – Atributos do elemento Layer

3.1. Atributo Queryable

O atributo booleano *queryable* indica se o servidor suporta a operação de GetFeatureInfo naquela Camada. Um servidor pode suportar GetFeatureInfo em algumas das suas camadas, mas não

necessariamente em todas. Um servidor emitirá uma excepção de serviço (código = "LayerNotQueryable") se GetFeatureInfo é pedido em uma Camada que não é *queryable*.

3.2. Atributo Cascaded

Uma Camada é classificada como "*cascaded*" se foi obtida a partir de um servidor diferente do qual a incluiu nos metadados de serviço. O segundo servidor pode simplesmente oferecer um ponto de acesso adicional para a Camada, oferecer formatos de output adicionais ou reprojecções para outros sistemas de referência de coordenadas (Open Geospatial Consortium, 2008).

Se um servidor WMS usar o conteúdo de outro WMS, então deverá incrementar o valor do atributo *cascaded* em uma unidade nas Camadas afectadas. Se esse atributo não existir nos metadados do serviço do servidor original, então o WMS inserirá o atributo com o seu valor igual a 1.

3.3. Atributo Opaque

Se o atributo booleano opcional *opaque* está ausente ou tem como valor *false*, então os mapas criados a partir daquela camada tem geralmente áreas sem dados significantes que um cliente pode exibir como transparente. Características vectoriais como pontos e linhas são consideradas não opacas neste contexto (embora a algumas escalas ou tamanhos de símbolo uma colecção de características pudesse encher a área de mapa). Um valor verdadeiro para *opaque* indica que a camada representa uma cobertura de área totalmente preenchida. A declaração *opaque* deveria ser levada como uma sugestão ao cliente para colocar tal camada no fundo de uma pilha de mapas.

Este atributo descreve o conteúdo de dados da camada, não o formato da imagem da resposta de mapa. Seja ou não uma camada opaca, um servidor deve ainda obedecer ao parâmetro TRANSPARENT relativo ao GetMap: quer dizer, o servidor enviará uma imagem com um fundo transparente se e só se o cliente pede TRANSPARENT=TRUE e um FORMAT de imagem que suporta transparência (Open Geospatial Consortium, 2008).

3.4. Atributo noSubsets, fixedWidth e fixedHeight

Os metadados de uma camada podem incluir três atributos opcionais que indicam se o servidor de mapas é menos funcional que um WMS normal, isto porque pode não extrair um subconjunto de um dataset maior ou porque só serve mapas de um tamanho fixo e não os pode redimensionar. Por exemplo, um WMS que aloja uma colecção de imagens digitalizadas de mapas históricos ou imagens de satélite pré-computadas, pode não ser capaz de obter subconjuntos ou redimensionar essas imagens. Porém, pode responder a pedidos de GetMap para mapas completos no tamanho original.

As colecções de imagens estáticas podem não ter um sistema de referência de coordenadas bem definido. Quando noSubsets é igual a *true* indica que o servidor não pode fazer um mapa de uma área geográfica diferente da bounding box da camada.

Quando presente e diferente de 0, fixedWidth e fixedHeight indicam que o servidor não pode produzir um mapa da camada a uma largura e altura diferente dos tamanhos fixos que indicou.

3.5. Elemento Title

O elemento <Title> é obrigatório em todas as camadas e define um texto humano-legível para apresentação num catálogo. O Título não é herdado pelas sub-camadas.

3.6. Elemento Name

Uma camada só pode ser requisitada a um servidor se e só se declara um <Name>, valor este que deve ser usado no parâmetro de LAYERS do pedido GetMap.

Um servidor lançará uma excepção de serviço (code="LayerNotDefined") se uma camada inválida é pedida.

Uma categoria pode por si própria incluir um <Name> pelo qual um mapa, que contém também todas as sub-camadas pode ser pedido reduzindo deste modo a quantidades de mapas a transferir e a quantidade de processamento a usar para apresentar todas as imagens ao mesmo tempo. Por exemplo, uma camada "Estradas" podem ter sub-camadas como "Auto-estradas" e "Estradas Nacionais" e permitir assim ao utilizador pedir individualmente ou todas as camadas de uma só vez.

O Name não é herdado pelas sub-camadas.

3.7. Elementos Abstract e KeywordList

Os elementos <Abstract> e <KeywordList> são opcionais, mas um servidor deveria implementa-los. O elemento Abstract é uma descrição narrativa sobre a camada, enquanto o elemento KeywordList poderá conter elementos para ajudar em procuras de catálogos. Estes elementos não são herdados pelas sub-camadas.

3.8. Elemento Style

Podem ser anunciados zero ou mais Styles para uma camada ou colecção de camadas usando elementos <Style> onde cada um terá elementos <Name> e <Title>. O <Name> do estilo é especificado no pedido através do parâmetro STYLES. O Title é uma string humano-legível. Se existe apenas um estilo disponível, este fica definido como o estilo por defeito e não precisa de ser anunciado.

O elemento <Style> pode conter vários outros elementos. O elemento <Abstract> contém uma descrição narrativa enquanto <LegendURL> contém o local de uma imagem para a legenda do mapa relativa ao estilo. O elemento <Format> dentro de <LegendURL> indica o MIME type da imagem de legenda, e os atributos opcionais *width* e *height* o tamanho da imagem em pixel. Os servidores devem indicar a largura e a altura se estes são conhecidos na hora de processar o pedido de GetCapabilities. A imagem de legenda deve identificar os símbolos, linhas e cores usadas na imagem do mapa. A imagem de legenda não deverá conter texto que duplica o Título da camada, porque essa informação já é conhecida pelo cliente e pode ser mostrada ao utilizador através de outros meios.

As declarações de Style são herdadas pelas sub-camadas. Uma sub-camada não pode redefinir um *Style* com o mesmo *Name* que o herdado da camada pai. Contudo uma sub-camada pode definir um Style novo com um *Name* novo que não está disponível para a camada de pai.

3.9. Elemento EX_GeographicBoundingBox

Toda a camada deverá ter exactamente um elemento <EX_GeographicBoundingBox> que pode ser explicitamente declarado ou herdado de uma camada de pai.

<EX_GeographicBoundingBox> declara através dos quatro elementos *westBoundLongitude, eastBoundLongitude, southBoundLatitude* e *northBoundLatitude*, o rectângulo mínimo que delimita em graus decimais a área coberta pela camada. <EX_GeographicBoundingBox> deverá estar presente independentemente do CRS que o servidor possa suportar, por isso deve ter-se em consideração que este pode ser aproximado, caso os dados não estejam nativamente em coordenadas geográficas. O propósito do EX_GeographicBoundingBox é facilitar procuras geográficas sem requerer uma reprojecção das coordenadas por parte da máquina de procura (Open Geospatial Consortium, 2008).

3.10. Elemento CRS

Todas as camadas têm pelo menos um CRS disponível. De modo a indicar quais os CRSs disponíveis para cada camada, toda a camada terá pelo menos um elemento <CRS> que ou é explicitamente declarado ou herdado de uma camada de pai. O elemento raiz <Layer> incluirá uma sucessão de zero ou mais elementos de CRS que listam todos os CRSs comuns a todas as sub-camadas. Uma sub-camada pode acrescentar opcionalmente mais elementos à lista herdada de uma camada pai. Qualquer duplicação será ignorada pelos clientes.

Quando uma camada estiver disponível em vários sistemas de referência de coordenada, a lista de valores de CRS disponíveis será representada como uma lista/sucessão de elementos <CRS> onde cada um contém um nome único de CRS.

Exemplo:

```
<CRS> CRS:84 </CRS>
<CRS> EPSG:26718 </CRS>
```

3.11. Elemento BoundingBox

Nos metadados do serviço WMS deve ser declarada uma ou mais BoundingBox para cada camada.

Um elemento BoundingBox pode ser declarado explicitamente ou pode ser herdado de uma camada pai. No XML, o elemento <BoundingBox> inclui os seguintes atributos:

- *CRS* indica o CRS que se aplica a esta BoundingBox.

- miny, minx, maxx, maxy indicam os limites da BoundingBox, e usa as unidades e ordem do eixo especificado no CRS.
- *resx* e *resy* (opcional) indicam a resolução espacial dos dados incluídos na camada nessas mesmas unidades.

Assim sendo, uma camada não deve conter uma BoundingBox para um CRS que não suporta. Reciprocamente, uma camada pode suportar CRSs para o qual não especifica uma BoundingBox.

Por outro lado, o parâmetro BBOX especifica qual a área que será retratada no mapa. A área de BBOX pode ou não sobrepor, conter ou ser contida dentro da área de BoundingBox.

Uma camada pode ter múltiplos elementos de BoundingBox, mas cada um declarará um CRS diferente. Uma camada herda qualquer valor de BoundingBox definido pelo seu pai. Um BoundingBox herdado para um CRS particular é substituído por qualquer declaração para o mesmo CRS na sub-camada. Um BoundingBox para um CRS novo que não tenha sido declarado pelo pai é acrescentado à lista de BoundingBox da camada (Open Geospatial Consortium, 2008).

O elemento <EX_GeographicBoundingBox> é conceptualmente semelhante a um BoundingBox no qual o atributo CRS = "CRS:84" está definido implicitamente. Porém, <EX_GeographicBoundingBox> não será usado como um substituto para <BoundingBox CRS = "CRS:84">. Se o servidor desejar fornecer informação de Bounding Box no CRS CRS:84, então deverá ser incluído explicitamente nos metadados do serviço um elemento de BoundingBox com o CRS:84.

3.12. Denominadores de escala

Os elementos <MinScaleDenominator> e <MaxScaleDenominator> definem os limites das escalas para os quais é apropriado gerar um mapa de uma camada.

Ao receber um pedido para um mapa que não está dentro dos limites dos denominadores de escala, o servidor pode devolver um mapa em branco, uma imagem da camada que está sobrecarregada com características ou no caso contrário pobremente revestida para visualização. Nesta situação o servidor não deverá responder com uma excepção de serviço (Open Geospatial Consortium, 2008).

Neste Padrão Internacional, o tamanho de um pixel comum é definido para ser de 0,28 mm x 0,28 mm.

Porque clientes arbitrários podem pedir mapas a um servidor, o verdadeiro tamanho do pixel do dispositivo de rendering final é desconhecido pelo servidor.

Os elementos MinScaleDenominator e MaxScaleDenominator, e como sugerem os nomes, definem o limite dos denominadores de escala de mapas para os quais uma camada é apropriada. A escala mínima é inclusiva e a escala máxima é exclusiva (Open Geospatial Consortium, 2008).

Exemplo:

```
<MinScaleDenominator>1e3</MinScaleDenominator>
<MaxScaleDenominator>1e6</MaxScaleDenominator>
```

3.13. Dimensões de amostra

O elemento opcional <Dimension> inclui os metadados necessários para dados multidimensionais.

As declarações de dimensão são sempre herdadas da camada pai. Qualquer declaração de Dimensão nova na camada com o mesmo nome da herdada do pai substitui o valor declarado pelo pai.

Por exemplo, consideremos um servidor que oferece imagens diárias de satélite da Terra. O servidor poderia dar a cada camada (dia) um nome diferente, e um cliente pediria o tempo usando o nome apropriado no pedido de GetMap. O número de camadas nos metadados do serviço iria aumentar diariamente, o que poderia piorar ainda mais se tivermos em conta outros factores como o comprimento de onda. Deste modo o servidor pode declarar um nome único para a camada e enumerar tempos disponíveis e comprimento de onda nos metadados do serviço. Um cliente terá de acrescentar parâmetros adicionais ao GetMap para obter um tempo e faixa específico.

Campo	Obrigatório	Significado
Name	Sim	Atributo que declara o nome do eixo dimensional. Deve ser sensível a maiúsculas/minúsculas e não deveria conter nenhum espaço branco.
Units	Sim	Atributo que indica as unidades de eixo dimensional. Se não existe nenhuma unidade usa-se a string nula: units =" ".
unitSymbol	Não	Atributo que especifica o símbolo.
Default	Não	Atributo que indica o valor por defeito que será usado se o pedido de GetMap não especificar um valor para a dimensão. Se o atributo estiver ausente, então responderá com uma excepção de serviço.
multipleValues	Não	Atributo booleano que indica se múltiplos valores da dimensão podem ser pedidos em simultâneo. 0 (ou "false") para valores únicos apenas; 1 (ou "true") de modo a permitir valores múltiplos. Por defeito o valor é 0.
nearestValue	Não	Atributo booleano que indica se o valor para a dimensão pode ser o valor mais próximo ao estipulado num pedido, ou se tem que ser exactamente o valor definido no pedido. Se 0 (ou "false"), o valor no pedido deve corresponder exactamente ao valor declarado; Se 1 (ou "true") o valor no pedido pode ser aproximado. Por defeito o valor é 0.
Current	Não	Atributo booleano válido só para extensões temporais (se o nome do atributo = "time"). Este atributo, se igual a "true" ou 1, indica que aqueles dados temporais normalmente são mantidos actualizados e que o parâmetro TIME num pedido pode incluir "current" como palavra-chave em vez do último valor. Por defeito é 0.
Extent	Sim	Texto que indica os valores disponíveis para a dimensão. A sintaxe para Extent é descrita na Tabela 4.

Tabela 3 – Atributos do elemento Dimension

Sintaxe	Descrição
Valor	Um valor único.
valor1, valor2,valor3,... [1]	Uma lista de valores múltiplos.
min/max/resolução	Um intervalo definido pelo seu valor inferior e superior e a sua resolução.
min1/max1/res1,min2/max2/res2,... [1]	Uma lista de intervalos múltiplos.
1) Espaços brancos são permitidos seguidos de uma vírgula em listas no elemento <Dimension> nos metadados do serviço.	

Tabela 4 - Sintaxe para valores de Extent

Um elemento Dimension tem o seguinte formato em XML:

```
<Dimension name="dimension_name" units=unit_name unitSymbol="symbol" default="default_value" multipleValues="0|1" nearestValue="0|1" current="0|1">extent</Dimension>
```

Exemplos da aplicação do elemento Dimension:

```
<Dimension name="time" units="ISO8601" default="2003-10-17">1996-01-01/2003-10-17/P1D</Dimension>
<Dimension name="temperature" units="Kelvin" unitSymbol="K" default="300">230,300,400</Dimension>
<Dimension name="elevation" units="CRS:88" unitSymbol="m" default="0">0/10000/100</Dimension>
```

No pedido é necessário adicionar um parâmetro cujo nome é construído através da concatenação do prefixo "DIM_" com o nome da dimensão em questão. O "DIM_nome" resultante não é sensível a maiúsculas/minúsculas. No caso das dimensões "time" e "elevation" não se usa o prefixo (Open Geospatial Consortium, 2008).

Exemplo:

```
<Dimension name="wavelength" units="Angstrom" unitSymbol="Ao">1000,2000,3000,4000<Dimension>.
```

Assim, o pedido de GetMap poderia incluir o parâmetro "DIM_WAVELENGTH=3000"

Se o valor de dimensão é inválido ou inexistente no pedido de um cliente, e nenhum valor por defeito ou valor aproximado for habilitado, então o servidor responderá com uma excepção de serviço (código=InvalidDimensionValue ou código=MissingDimensionValue, como apropriado).

3.14. Elemento MetadataURL

Um servidor deveria usar um ou mais elementos <MetadataURL> para oferecer metadados padronizados e detalhados, sobre os dados que correspondem a uma camada particular. O atributo de "type" indica o padrão usado nos metadados. Estão definidos dois valores para o atributo "type" por este Padrão Internacional: o valor que "ISO 19115:2003" que corresponde ao ISO 19115:2003 e o valor "FGDC:1998" que corresponde ao FGDC-STD-001-1998. O elemento incluso <Format> indica o formato de arquivo MIME type do registo dos metadados (Open Geospatial Consortium, 2008).

Os elementos MetadataURL não são herdados pelas sub-camadas.

3.15. Elemento Attribution

O elemento opcional <Attribution> providencia um modo para identificar a fonte da informação geográfica usada em uma camada ou colecção de camadas. Este elemento inclui vários sub-elementos opcionais:

- <OnlineResource> declara o URL do fornecedor de dados;
- <Title> é um texto humano-legível com o nome do fornecedor de dados;
- <LogoURL> é o URL de uma imagem de logótipo. As aplicações cliente podem escolher exibir um ou mais destes itens. Um elemento <Format> em LogoURL indica o tipo da imagem do logótipo e os atributos width e height o tamanho da imagem em pixel.

O elemento Attribution é herdado através de sub-camadas. Qualquer redefinição pela sub-camada substitui o valor que foi herdado.

3.16. Elementos Identifier e AuthorityURL

Um servidor pode usar zero ou mais elementos <Identifier> para listar números de ID ou rótulos definidos por uma Autoridade particular.

O conteúdo do elemento <Identifier> é o valor de ID. O atributo authority do elemento Identifier corresponde ao atributo *name* de um elemento <AuthorityURL>.

O AuthorityURL inclui um elemento <OnlineResource> que declara o URL de um documento que define o significado dos valores no Identifier.

Exemplo:

```
<AuthorityURL name="algum_nome">
    <OnlineResource xlink:href="algum_url" ... />
</AuthorityURL>
<Identifier authority="algum_nome">valor_id</Identifier>
```

O AuthorityURL é herdado pelas sub-camadas. Uma sub-camada não pode definir um AuthorityURL com o mesmo atributo *name* herdado do pai. O elemento Identifier não é herdado. Uma camada não declarará um Identifier a menos que o AuthorityURL correspondente esteja previamente declarado ou tenha sido herdado.

3.17. Elemento FeatureListURL

Um servidor pode usar o elemento <FeatureListURL> para especificar uma lista das características representadas numa camada.

O FeatureListURL não é herdado pelas sub-camadas. Quando se inclui o elemento Format em FeatureListURL, este indica o formato de arquivo da lista de características.

3.18. Elemento DataURL

Um servidor pode usar DataURL para oferecer uma ligação aos dados subjacentes representados por uma camada particular. O elemento <Format> (se incluído) indica o formato de arquivo MIME type dos dados.

O DataURL não é herdado pelas sub-camadas.

3.19. Herança das propriedades de uma Camada

A Tabela 5 resume como as propriedades de uma <Layer> são transmitidas as sub-camadas. De um modo geral, as propriedade podem simplesmente não ser herdadas, herdadas como são, redefinidas se a sub-camada as redefine, ou herdadas e adicionadas se a sub-camada também as define. Na Tabela 5, a coluna *número* declara o número de vezes que cada elemento pode aparecer numa Camada, quer explicitamente ou através de herança. Os valores nesta coluna usam a seguinte notação:

- 1: aparece precisamente uma vez em cada Camada.
- 0/1: não aparece ou aparece uma vez.
- 0+: aparece zero ou mais vezes.
- 1+: aparece uma ou mais vezes.

A coluna *Herança* indica se, ou como o elemento é herdado sub-camadas. Os valores nesta coluna seguem a seguinte notação:

- Não: Não é herdado. Se o elemento é definido para ser obrigatório por este Padrão Internacional, então cada elemento Layer deve incluir aquele elemento.
- Substituição: O valor pode ser herdado do pai e ser omisso pelas sub-camadas, mas se é especificado pelas sub-camadas então o valor do pai é ignorado.
- Adição: O valor pode ser herdado do pai e ser omisso pelas sub-camadas. "Adição" só é pertinente para elementos que podem se aparecer mais de uma vez. As sub-camadas herdam

qualquer valor provido pelo pai e adicionam algum valor à sua própria lista. Qualquer definição duplicada pela sub-camada é ignorada.

Elemento	Número	Herança	Obrigatório
Layer	0+	Não	Não
Name	0/1 a)	Não	Não
Title	1	Não	Sim
Abstract	0/1	Não	Não
KeywordList	0/1	Não	Não
Style	0+	Adição	Não
CRS	1+ b)	Adição	Sim
EX_GeographicBoundingBox	1 b)	Substituição	Sim
BoundingBox	1+ b)	Substituição	Sim
Dimension	0+	Substituição	Não
Attribution	0/1	Substituição	Não
AuthorityURL	0+	Adição	Não
Identifier	0+	Não	Não
MetadataURL	0+	Não	Não
DataURL	0/1	Não	Não
FeatureListURL	0/1	Não	Não
MinScaleDenominator	0/1	Substituição	Não
MaxScaleDenominator	0/1	Substituição	Não

Tabela 5 - Propriedades do elemento Layer

4. Operações

O padrão WMS especifica três operações, as duas primeiras são obrigatórias e a terceira é opcional. Estas operações são apresentadas brevemente abaixo.

- GetCapabilities – Com este pedido, o cliente adquire uma descrição de todas as camadas oferecidas pelo servidor.
- GetMap – Este é a operação para pedir uma camada específica ao servidor.

- GetFeatureInfo – A operação é opcional. Com esta operação o cliente pode perguntar ao servidor por informações sobre características apresentadas no mapa.

4.1. WMS GetCapabilities

O pedido GetCapabilities permite obter a seguinte informação sobre um serviço de WMS:

- Todas as interfaces que o serviço de WMS pode suportar
- Formatos de imagem que pode produzir (por exemplo JPEG, PNG, GIF)
- Lista de sistemas de referência espaciais disponíveis para entrega de dados
- Lista de todos os formatos de excepções para retorno de excepções
- Lista de todas as capacidades vendedor-específicas (ou propriedades) que estão disponíveis para modificar ou controlar acções do serviço, cada uma com o seu valor actual
- Lista dos mapas (camadas) disponíveis no serviço
- Lista de camadas que são suportadas pela interface de *GetFeatureInfo* que é opcional

4.1.1. Pedido de GetCapabilities

Os parâmetros seguintes podem ser usados num pedido de GetCapabilities.

Parâmetro	Obrigatório	Descrição
VERSION=version	Não	Versão para o protocolo de comunicação.
REQUEST=GetCapabilities	Sim	Identificação do tipo de pedido.
SERVICE=WMS	Sim	Tipo de serviço. No caso deste protocolo o valor é sempre "WMS".
FORMAT=MIME_type	Não	Tipo de formato desejado para os dados de saída.
UPDATESEQUENCE=string	Não	Número de sequência ou string para controlo da cache.

Tabela 6 – Visão geral do pedido GetCapabilities

Este exemplo mostra o número mínimo de parâmetros necessários para efectuar um pedido de GetCapabilities válido:

```
http://servidor/caminho/script?VERSION=1.3.0&REQUEST=GetCapabilities&SERVICE=WMS&
```

Onde,

- Servidor – é o nome do domínio de URL para o site, como por exemplo www.google.com ou um nome qualificado do servidor.

- VERSION=1.3.0 – é a versão do pedido.
- REQUEST=GetCapabilities – especifica o nome da operação.

4.1.2. Resposta de GetCapabilities

Para além dos elementos especificados em 4 acima para a descrição do serviço, este serviço inclui ainda o elemento opcional <LayerLimit> que é um inteiro positivo que indica o número máximo de camadas que um cliente pode incluir em um único pedido de GetMap. Se este elemento estiver ausente, o servidor não impõe nenhum limite.

Os elementos opcionais <MaxWidth> e <MaxHeight> nos metadados do serviço são inteiros positivos que indicam a largura e altura máxima permitida, que um cliente pode incluir em um único pedido de GetMap. Se qualquer elemento está ausente o servidor não impõe nenhum limite no parâmetro correspondente.

Para além disso, contem no elemento <Capability> a lista de todas as camadas e estilos disponíveis.

Cada mapa disponível é descrito por um elemento <Layer> nos metadados do serviço. Conceptualmente, cada camada é uma entidade distinta. Porém, como um meio de classificar e organizar as camadas, e como um meio de reduzir o tamanho dos metadados do serviço, uma única camada pai pode incluir qualquer número de camadas adicionais que podem ser alinhadas hierarquicamente como se desejar. Algumas propriedades definidas em uma camada pai são herdadas pelas sub-camadas, e estas propriedades herdadas podem ser redefinidas ou serem simplesmente usadas. A subcláusula 3.19 acima resume como ou se a propriedade é herdada.

Um servidor deverá incluir pelo menos um elemento <Layer> para cada camada de mapa oferecida.

4.2. WMS GetMap

A resposta a um pedido GetMap válido é um mapa com as camadas pedidas, limitadas a área espacial desejada, nos estilos desejados, e tendo em conta o sistema de referência de coordenada que foi especificado, tamanho, formato e transparência.

Um pedido de GetMap inválido leva a uma produção de uma excepção de serviço (ou um erro no protocolo rede em casos extremos).

Parâmetro	Obrigatório	Descrição
VERSION=x.y.z	Sim	Versão do pedido
REQUEST=GetMap	Sim	Nome/tipo do pedido
LAYERS=lista_de_camadas	Sim	Lista separada por vírgulas de uma ou mais camadas que formam o mapa

Parâmetro	Obrigatório	Descrição
STYLES=lista_de_estilos	Sim	Lista separada por vírgulas de um ou mais estilos que correspondem as camadas que formam o mapa
CRS=namespace:identificador	Sim	Sistema de coordenadas de referência
BBOX=minx,miny,maxx,maxy	Sim	Bounding box (inferior esquerdo, superior direito) em unidades de CRS
WIDTH=output_largura	Sim	Largura em pixéls da imagem[2]
HEIGHT=output_alturar	Sim	Altura em pixel da imagem[2]
FORMAT=output_formato	Sim	Formato da imagem[2] produzida
TRANSPARENT=TRUE|FALSE	Não	Mapa com fundo transparente se o formato da imagem o suporta e tem o valor *true* (o valor por defeito é FALSE)
BGCOLOR=cor	Não	Valor hexadecimal RGB para a cor de fundo (default=0xFFFFFF)
EXCEPTIONS=formato_da_excepção	Não	O formato no qual excepções serão devolvidas pelo servidor WMS (default=XML).
TIME=tempo	Não	Valor de Tempo para a camada desejada, se aplicável
ELEVATION=elevação	Não	Elevação da camada desejada
Outras dimensões de amostra DIM_	Não	Valor de outras dimensões como apropriado

Tabela 7 - Visão geral do pedido GetMap

4.2.1. Parâmetros permitidos

4.2.1.1. VERSION & REQUEST

Tem o mesmo significado que em Anexo I.3.3 e Anexo I.3.2 acima.

4.2.1.2. LAYERS

O parâmetro obrigatório LAYERS lista as camadas a serem representadas no mapa a ser devolvido. O valor do parâmetro LAYERS é uma lista vírgula-separada de um ou mais nomes de camada válidos. Os nomes de camada permitidos são os existentes em qualquer elemento <Layer> <Name> nos metadados do serviço.

Um WMS processará o pedido começando com a camada mais a esquerda na lista.

[2] A imagem pode ser um vídeo caso seja esse o formato pedido pelo utilizador

4.2.1.3. STYLES

O parâmetro obrigatório STYLES lista o estilo com o qual cada camada será representada. O valor do parâmetro STYLES é uma lista separada por vírgulas de um ou mais nomes de estilo válidos. Há uma correspondência de um para um entre os valores no parâmetro de LAYERS e os valores no parâmetro STYLES. O nome do estilo a aplicar a camada deve estar definido num elemento <Style><Name> directamente contido ou herdado pela camada (elemento <Layer>) nos metadados do serviço. (Por outras palavras, o cliente pode não pedir uma camada em um estilo que só estava definido para uma camada diferente.) Um servidor lançará uma excepção de serviço (código = StyleNotDefined) se um estilo não definido é pedido. Um cliente pode pedir o estilo por defeito usando um valor nulo (como "STYLES="). Se são pedidas várias camadas com uma mistura de estilos entre o estilo por defeito e outros estilos, o parâmetro STYLES incluirá valores nulos entre vírgulas (como em "STYLES=style1,,style3,"). Se todas as camadas usam o estilo por defeito então "STYLES=" ou "STYLES=," é uma forma válida (Open Geospatial Consortium, 2008).

Se o servidor anuncia vários estilos para uma camada, e o cliente envia um pedido para o estilo por defeito, a escolha do estilo a usar por defeito está à descrição do servidor. A ordem dos estilos nos metadados do serviço não indica qual é o estilo por defeito.

Este padrão não permite que o utilizador defina o seu próprio estilo para o mapa, deixando assim o utilizador limitado aos estilos definidos no servidor. Esta limitação pode ser removida se para além do nome seja também possível incluir a definição do estilo. Os servidores já possuem todos os recursos para poderem suportar esta útil extensão, como será demonstrado na definição da arquitectura.

4.2.1.4. CRS

O parâmetro de pedido CRS declara o CRS que se aplica ao parâmetro BBOX. O valor do parâmetro de CRS num pedido para um servidor particular será um dos valores definidos nos metadados do serviço do servidor dentro um elemento <CRS> definido ou herdado pela camada pedida. O mesmo CRS aplica-se a todas as camadas existentes no pedido. Deve-se ao facto de não fazer sentido constituir uma imagem com CRS múltiplos, assim como derivados de um BBOX num CRS específico (Open Geospatial Consortium, 2008).

Um servidor de WMS não precisa de suportar todos os CRS possíveis, mas tem que declarar nos metadados do serviço os que suporta, e como tal terão que ser aceites nos pedidos. Se um pedido contém um CRS não oferecido pelo servidor, o servidor lançará uma excepção de serviço (código = "InvalidCRS").

Não é exigido que os clientes suportem todos os possíveis CRS. Se um cliente e servidor não conseguirem acordar um CRS, o cliente pode deixar de comunicar com o servidor e procurar um serviço intermediário que faça a reprojecção das coordenadas.

Este Padrão Internacional não define um método para os clientes pedirem explicitamente uma reprojecção ou transformação de coordenadas. Porém, a reprojecção pode acontecer implicitamente se o servidor oferecer múltiplos CRS e internamente guarda um CRS particular para a informação geográfica.

Se o servidor de WMS declarou CRS=CRS:1 para uma camada, então a camada não tem um sistema de referência de coordenadas bem definido e as unidades do parâmetro BBOX serão em pixéis.

4.2.1.5. BBOX

O parâmetro obrigatório BBOX permite a um cliente pedir uma Bounding Box particular. O valor do parâmetro BBOX num pedido GetMap é uma lista de números reais separados por uma vírgula (ver também 2 acima).

4.2.1.6. FORMAT

O parâmetro obrigatório FORMAT declara o formato desejado do mapa. São listados os valores suportados para um pedido GetMap em um ou mais elementos <Request><GetMap><Format> nos metadados do serviço. Não existe nenhum formato por defeito.

Em um ambiente de HTTP, o MIME type será escolhido para o objecto será usado como Content-type no cabeçalho da resposta do servidor.

Se o pedido especificar um formato não suportado pelo servidor, este emitirá uma excepção de serviço (código = InvalidFormat).

4.2.1.7. WIDTH, HEIGHT

Os parâmetros obrigatórios WIDTH e HEIGHT especificam o tamanho em pixéis (inteiro) do mapa a ser produzido. O WIDTH-1 especifica o valor máximo no eixo *i* no CS do Mapa, e HEIGHT-1 especifica o valor máximo no eixo de *j* no CS do Mapa.

Se o pedido é para um formato de imagem, a imagem devolvida, independentemente do seu MIME type, terá exactamente a largura e altura especificada em pixéis. No caso em que a razão do aspecto da BBOX e a razão largura / altura seja diferente, o WMS estirará (stretch) o mapa devolvido. Em outras palavras, será possível usando esta definição pedir um mapa para um dispositivo onde as suas dimensões em pixéis não representam um quadrado ou estirar um mapa em uma área de imagem de uma razão de aspecto diferente (Open Geospatial Consortium, 2008).

Serão introduzidas distorções no mapa se a razão WIDTH / HEIGHT não for proporcional com *X*, *Y* e a razão aspecto de pixéis. Na programação de um cliente deve ter-se em conta a razão WIDTH/HEIGTH de modo a minimizar a possibilidade de os utilizadores pedirem inadvertidamente ou inconscientemente mapas torcidos.

Se um pedido for para um formato de elemento gráfico que não tem largura e altura explícita, o cliente deve incluir os valores correctos para WIDTH e HEIGHT no pedido, e um servidor pode os usar como informação útil na produção do mapa.

Os elementos opcionais <MaxWidth> e <MaxHeight> nos metadados do serviço são inteiros que indicam a largura e a altura máxima que é permitida a um cliente incluir num único pedido de GetMap.

Se qualquer elemento estiver ausente, o servidor não impõe nenhum limite no parâmetro correspondente.

Se o servidor de WMS declarou uma camada com largura e altura fixas, então o cliente especificará exactamente essa WIDTH e HEIGHT no pedido, ou o servidor poderá emitir uma excepção de serviço.

4.2.1.8. TRANSPARENT

O parâmetro opcional TRANSPARENT especifica se o fundo de mapa é transparente ou não.

TRANSPARENT pode assumir dois valores, "TRUE" ou "FALSE". O valor por defeito é FALSE caso este parâmetro esteja ausente no pedido.

A habilidade para devolver imagens com transparências permite a sobreposição de mapas diferentes. É fortemente recomendado que todo o WMS ofereça um formato que suporte transparência para as camadas que poderiam ser aplicadas acima de outras[3].

Quando o parâmetro TRANSPARENT tem o valor TRUE e o parâmetro FORMAT contém um formato de Imagem (por exemplo image/gif), o serviço devolverá (se permitido pelo formato do pedido) um resultado onde todos os pixéis que não representam características ou valores de dados naquela camada, são definidos como um valor transparente. Por exemplo, uma camada de "estradas" seria transparente onde nenhuma estrada é desenhada. Se o formato de imagem não suportar transparência, então o servidor responderá com uma imagem opaca (em outras palavra, não é um problema para o cliente pedir sempre mapas transparentes independente do formato). Quando TRANSPARENT tem o valor FALSO, os pixéis sem dados serão definidos com a cor do parâmetro BGCOLOR (veja mais a frente).

Quando uma camada é declarada "opaque", então porções significantes, ou a totalidade do mapa não contem zonas transparentes, contudo os clientes podem ainda pedir TRANSPARENT=true.

4.2.1.9. BGCOLOR

O parâmetro opcional BGCOLOR é uma string que especifica a cor a ser usada nos pixéis de fundo (pixéis que não representam dados) do mapa. O formato geral de BGCOLOR é uma codificação hexadecimal de um valor RGB (O formato é 0xRRGGBB; O prefixo de "0x" terá um "x" minúscula) onde os valores para vermelho (R), verde (G) e azul (B) podem variar entre 00 e FF (0 e 255, base 10). O valor por defeito é 0xFFFFFF (correspondendo à cor branco) se este parâmetro está ausente no pedido.

Se uma camada foi declarada como "opaque" então porções significantes ou a totalidade do mapa não pode apresentar qualquer fundo personalizado.

4.2.1.10. EXCEPTIONS

[3] O formato image/gif suporta transparência e é exibido correctamente por clientes web comuns. O formato image/png suporta uma gama de opções de transparência mas o seu suporte em aplicações é menos comum. O formato de image/jpeg não suporta transparência

No caso do serviço WMS alguns dos códigos de excepção devolvidos são apresentados na Tabela 8.

Código de Excepção	Significado
InvalidFormat	O pedido exige um formato não suportado pelo servidor.
InvalidCRS	O pedido referência um CRS não é suportado pelo servidor para uma ou mais camadas requisitadas no pedido.
LayerNotDefined	O pedido GetMap inclui uma camada que o servidor não conhece, ou o pedido GetFeatureInfo inclui uma camada que não está presente no mapa.
StyleNotDefined	Pedido de uma camada num estilo não oferecido pelo servidor.
LayerNotQueryable	O pedido GetFeatureInfo está a ser aplicado a uma camada que não é declarada como *queryable*.
InvalidPoint	O pedido GetFeatureInfo contém um valor inválido para I ou J.
CurrentUpdateSequence	O valor (opcional) do parâmetro UpdateSequence no pedido de GetCapabilities é igual ao valor actual da sequência update nos metadados do serviço.
InvalidUpdateSequence	O valor (opcional) do parâmetro UpdateSequence no pedido de GetCapabilities é maior que o valor actual da sequência update nos metadados do serviço.
MissingDimensionValue	O pedido não inclui um valor de dimensão de amostra e o servidor não declarou um valor por defeito para a dimensão.
InvalidDimensionValue	O pedido contém um valor de dimensão de amostra inválido.
OperationNotSupported	O pedido é para uma operação opcional que não é suportada pelo servidor.

Tabela 8 - Códigos de excepções de serviço

Para mais informações sobre este parâmetro consultar Anexo I, item 3.7.

4.2.1.11. TIME

Parâmetro que especifica o tempo ou do intervalo de tempo desejado (Exemplo: TIME=2000-08-03, Filme: TIME=2000-07-01/2000-07-31/P1D).

4.2.1.12. ELEVATION

Parâmetro que especifica a elevação desejada (exemplo: ELEVATION=1000).

4.2.1.13. Dimensões de amostra

Parâmetros que especificam os valores desejados para as diversas dimensões de amostra. Para mais informações veja o item Anexo II, item 3.13.

4.3. WMS GetFeatureInfo

GetFeatureInfo é uma operação opcional. Só é suportada para camadas para qual o atributo queryable = "1" esteja definido ou herdado. Um cliente não emitirá um pedido de GetFeatureInfo para outras camadas. Um WMS responderá com uma resposta de excepção (XML) de serviço (código = OperationNotSupported) correctamente formatada se receber um pedido de GetFeatureInfo mas não o suporta.

A operação de GetFeatureInfo é desenhada para proporcionar aos clientes de um WMS mais informação sobre as características em imagens de mapas que foram devolvidos através de pedidos de GetMap prévios. O caso de uso comum para GetFeatureInfo é o de um utilizador que após a resposta de um pedido de GetMap escolhe um ponto (I, J), possivelmente através do uso do rato, para obter mais informação sobre essa localização. A operação básica provê a habilidade de um cliente especificar qual é o pixel em questão para a camada e o formato em que a informação deveria ser devolvida (Open Geospatial Consortium, 2008).

As semânticas de como um WMS decide devolver informação sobre isso ou o que devolver exactamente, fica ao encargo do serviço de WMS.

O servidor devolverá uma resposta de acordo com o pedido INFO_FORMAT se o pedido é válido, ou emitirá uma excepção de serviço caso contrário. A natureza da resposta está à descrição da implementação do serviço, mas pertencerá à característica mais próxima de (I, J).

4.3.1. Parâmetros para a operação GetFeatureInfo

A Tabela 9 apresenta um resumo dos parâmetros

Parâmetro	Obrigatório	Descrição
VERSION=1.3.0	Sim	Versão do pedido
REQUEST=GetFeatureInfo	Sim	Nome da operação
Parâmetros de GetMap	Sim	Representa uma sucessão de parâmetros do pedido GetMap que gerou o mapa original. São omissos dois dos parâmetros de GetMap porque GetFeatureInfo especifica os seus próprios valores: VERSION e REQUEST
QUERY_LAYERS=lista_camadas	Sim	Lista de uma ou mais camadas a analisar no pedido.
INFO_FORMAT=formato_output	Sim	Formato em que a informação será devolvida (MIME type).

Parâmetro	Obrigatório	Descrição
FEATURE_COUNT=número	Não	Numero de características a descrever (default=1).
I=pixel_coluna	Sim	Coordenada *i* em pixéis da característica no mapa.
J=pixel_linha	Sim	Coordenada *j* em pixéis da característica no mapa.
EXCEPTIONS=exception_format	Não	Formato no qual as excepções são devolvidas (default= XML).

Tabela 9 - Parâmetros possíveis na operação GetFeatureInfo

4.3.1.1. VERSION & REQUEST

Tem o mesmo significado que em Anexo I.3.3 e Anexo I.3.2 acima.

4.3.1.2. QUERY_LAYERS

O parâmetro obrigatório QUERY_LAYERS declara as camadas que se pretendem analisar. O seu valor é uma lista separada por vírgulas de uma ou mais camadas. Este parâmetro conterá pelo menos um nome de uma camada, e pode conter menos camadas que o pedido GetMap original.

Se qualquer camada no parâmetro de QUERY_LAYERS não estiver definida nos metadados do serviço do WMS, o servidor emitirá uma excepção de serviço (código = LayerNotDefined).

4.3.1.3. INFO_FORMAT

O parâmetro obrigatório INFO_FORMAT indica o formato a usar ao devolver a informação sobre as características.

Os valores suportados para um pedido de GetFeatureInfo em um servidor de WMS são listados como MIME types em um ou mais elementos <Request> <FeatureInfo> <Format> nos metadados do serviço. Se o pedido especificar um formato que não é suportado pelo servidor, o servidor emitirá uma excepção de serviço (código = InvalidFormat).

EXEMPLO: O parâmetro INFO_FORMAT=text/xml requer que a informação sobre as características seja formatada em XML.

4.3.1.4. FEATURE_COUNT

O parâmetro opcional FEATURE_COUNT declara o número máximo de características a devolver por camada. O seu valor é um inteiro positivo. O valor por defeito é 1 caso o parâmetro esteja ausente ou seja diferente de um inteiro positivo.

4.3.1.5. I, J

Os parâmetros obrigatórios I e J são inteiros que indicam um ponto de interesse no mapa que foi produzido pelo pedido de GetMap embutido neste pedido. O ponto (I, J) é um ponto no espaço (i, j) definido pelo Mapa CS. Por essa mesma razão:

- O valor de *I* estará entre 0 e o valor de máximo do eixo *i*;
- O valor de *J* estará entre 0 e o valor de máximo do eixo *j*;
- O ponto *I*=0, *J*=0 indica o pixel ao canto esquerdo superior do mapa;
- *I* aumenta para a direita e *J* aumenta para baixo.

O ponto (I, J) representa o centro do pixel indicado. Se o valor de I ou J é inválido, o servidor emitirá uma excepção de serviço (código = InvalidPoint).

4.3.1.6. EXCEPTIONS

O parâmetro opcional EXCEPTIONS está definido em Anexo I, item 3.7 e a lista de códigos de excepção está definida em Anexo II, item 4.2.1.10. Se este parâmetro estiver ausente no pedido, o valor por defeito é "XML".

Anexo III. Web Feature Service (WFS)

Um OGC WFS pode ser descrito como: "Um serviço cujo comportamento segue uma interface de especificação que descreve os pedidos e respostas a serem efectuados por um servidor de características padrão" (ESRI, 2003).

O protocolo WMS permite a um cliente obter imagens de mapas de múltiplos serviços WMS. De modo semelhante, o WFS permite a um cliente obter, adicionar ou actualizar dados georreferenciados, codificados em GML em múltiplos serviços WFS.

Os requerimentos para um servidor WFS são:

- As interfaces devem ser definidas em XML.
- GML deve ser usado para expressar características dentro da interface.
- O mais básico WFS deve poder apresentar características usando GML.
- A fonte de armazenamento de características geográficas deve ser opaca a aplicações cliente e o único método de acesso aos dados deve ser pela interface de WFS.
- O uso de um subconjunto de expressões de XPath para propriedades de referenciamento.

Através de um servidor WFS pode ser possível invocar as seguintes operações ou elementos usando a plataforma de computação distribuída HTTP:

- INSERT – Criar uma nova característica
- DELETE – Eliminar uma ou mais características existentes no serviço
- UPDATE – Actualizar alguma propriedade de uma ou mais características existentes no serviço
- LOCK – Criar uma fechadura numa característica que vai ser editada de modo a manter a consistência. Assim não é possível que dois utilizadores diferentes editem a mesma característica ao mesmo tempo.
- DISCOVER e QUERY – Descobrir e interrogar o serviço por características com base em restrições espaciais ou não espaciais.

Pode-se classificar o WFS em três tipos: WFS básico, WFS XLink e WFS Transacção (WFS-T). O WFS básico é um serviço somente de leitura onde os clientes não podem modificar nenhuma característica servida pelo servidor. O WFS básico tem que suportar as seguintes três operações:

- GetCapabilities – Quando um WFS receber este pedido, responde com uma descrição das suas capacidades.
- DescribeFeatureType – Operação na qual o WFS descreve a estrutura de uma ou mais características.
- GetFeature – Com este pedido, o cliente pede características ao serviço. O cliente deve ser capaz de constranger a questão e especificar as propriedades da característica em que está interessado.

Com WFS-T (WFS de transacção) é possível para o cliente executar operações como criar, actualizar e apagar características. O WFS-T tem que suportar as três operações básicas, e a operação de transacção adicional (Open Geospatial Consortium, 2008).

- GetFeatureWithLock – Este elemento é funcionalmente semelhante ao elemento de GetFeature, excepto que indica a um WFS para tentar bloquear as características que são seleccionadas; presumivelmente para actualizar as características em uma operação subsequente (Open Geospatial Consortium, 2008).
- LockFeature – Expõe um mecanismo de travamento a longo prazo sobre uma característica para assegurar consistência (Opcional).
- Transaction – Usado para descrever operações de transformação de dados, que serão aplicadas a instâncias de característica acessíveis na web (Open Geospatial Consortium, 2008).

Exemplo de uma transacção WFS:

```
<?xmlversion="1.0"?>
<wfs:Transaction version="1.0.0" service="WFS" ....>
    <LockId>0D1</LockId>
    <wfs:Insert>... GML ... </wfs:Insert>
    <wfs:Update typeName="myns:RIVERS">
        <wfs:Property>
            <wfs:Name>myns:DEPTH</wfs:Name>
            <wfs:Value>150</wfs:Value>
        </wfs:Property>
        <ogc:Filter>
            <ogc:PropertyIsGreaterThan>
                <ogc:PropertyName>DEPTH</ogc:PropertyName>
                <ogc:Literal>100</ogc:Literal>
            </ogc:PropertyIsGreaterThan>
        </ogc:Filter>
    </wfs:Update>
    <wfs:DeletetypeName="myns:RIVERS">
        <ogc:Filter>
            <ogc:FeatureIdfid="RIVER.1013"/>
        </ogc:Filter>
    </wfs:Delete>
</wfs:Transaction>
```

Um WFS XLink suporta todas as operações de um servidor WFS básico e além disso implementa a operação de GetGmlObject para XLinks locais e/ou remotos e oferece a opção da operação de GetGmlObject ser executada durante operações de GetFeature.

WFS de Transacção

Um WFS de transacção suporta todas as operações de um WFS básico e além disso implementa as operações de transacção. Opcionalmente, um WFS transaccional pode implementar o GetGmlObject e/ou operações de LockFeature.

O processamento de pedidos procederia como se segue:

1) Uma aplicação cliente pediria o documento de capacidades do WFS. Tal documento contém uma descrição de todas as operações que o WFS suporta e uma lista de todos os tipos de características que pode fornecer.
2) Uma aplicação cliente (opcionalmente) faz um pedido ao servidor WFS para obter a definição de uma ou mais características que o WFS pode fornecer.
3) Baseado na definição do tipo de característica, a aplicação cliente gera o respectivo pedido.
4) O pedido é enviado a um servidor web.
5) O WFS descodifica e processa o pedido.
6) Quando o WFS termina de processar o pedido, gera um relatório de estado que devolverá ao cliente. No caso em que ocorram erros, o relatório de estado indicará essa informação.

1. Suporte de múltiplos namespaces

Um documento de esquema XML pode descrever apenas elementos que pertencem a um único namespace. Isto significa que um WFS não pode descrever características de namespaces múltiplos em um único documento. Para superar esta limitação, um WFS pode gerar um documento de esquema XML que é um "wrapper" que importa os esquemas de características de vários namespaces no pedido (Open Geospatial Consortium, 2008).

2. Parâmetros comuns

A tabela seguinte descreve os parâmetros comuns a todos os pedidos de WFS.

Parâmetro	Obrigatório	Valor por defeito	Descrição
VERSION	Sim	A maior das versões	Versão do pedido
SERVICE	Sim		Tipo de serviço
REQUEST	Sim		Nome da operação

Parâmetro	Obrigatório	Valor por defeito	Descrição
NAMESPACE	Não		Usado para especificar um namespace e o seu prefixo. O formato deve ser xmlns (prefix=escaped_url). Se o prefixo não é especificado então este pertence ao namespace por defeito. Mais do que um namespace podem ser ligados por uma lista específica, separada por vírgulas, de valores de xmlns ().
Parâmetros adicionais	Não		Possíveis parâmetros adicionais
Parâmetros Vendedor-específicos	Não		Parâmetros opcionais vendedor-específicos

Tabela 10 – Parâmetros WFS comuns

3. Domínios de parâmetros e Restrições

Os elementos <ows:Parameter> e <ows:Constraint> estão definidos na Especificação de Implementação Comum OWS e permite que valores válidos do domínio e restrições possam ser definidos globalmente para todas as operações ou localmente para operações específicas que um WFS oferece.

A tabela seguinte define os domínios de parâmetro que podem ser definidos no documento de capacidades de um WFS.

Nome da Operação	Nome do parâmetro
Todas as operações (Globalmente)	SrsName
GetCapabilities	AcceptVersions
	AcceptFormats
	Sections
DescribeFeatureType	OutputFormat
GetFeature	ResultType
	OutputFormat
GetFeatureWithLock	ResultType
	OutputFormat
GetGMLObject	OutputFormat
LockFeature	LockAction

Nome da Operação	Nome do parâmetro
Transaction	InputFormat
	Idgen
	ReleaseAction

Tabela 11 - Dominio de parâmetros que podem ser definidos

A tabela seguinte define as restrições que podem ser especificadas por um WFS no seu documento de capacidades.

Nome da restrição	Descrição
SupportsSOAP	Especifica que o serviço pode processar pedidos usando HTTP POST e dentro de um envelope SOAP
DefaultMaxFeatures	Especifica o valor por defeito para o atributo maxFeatures do elemento <GetFeature>. Se a restrição não é especificada então não existe limite no número de características que um pedido de GetFeature pode devolver.
LocalTraverseXlinkScope	Define o número mínimo e máximo de níveis locais que um WFS tentará resolver. O valor * significa todos os níveis e é o valor por defeito.
RemoteTraverseXlinkScope	Define o número mínimo e máximo de níveis remotos que um WFS tentará resolver. O valor * significa todos os níveis e é o valor por defeito.
DefaultLockExpiry	Define o tempo por defeito para que uma fechadura expire (em minutos). Se a restrição não é especificada então as fechaduras serão mantidos indefinidamente e exigirão a intervenção do administrador para os remover.

Tabela 12 - Operações de restrição

Ver exemplo em Anexo VIII.

4. Operações

De modo a suportar transacções e processar pedidos, estão definidas as operações que se seguem.

4.1. GetCapabilities

Um WFS deve ser capaz de descrever as suas capacidades. Especificamente, tem que indicar os tipos de características que pode fornecer e as operações que são suportadas em cada tipo de característica.

4.1.1. Pedido de GetCapabilities

Os parâmetros que podem ser usados num pedido de GetCapabilities são os mesmos que podem ser usados no serviço WMS (ver Anexo II).

Exemplo:

---

http://www.servidor.com/script?VERSION=1.3.0&SERVICE=WFS&REQUEST=GetCapabilities&

4.1.2. Resposta de GetCapabilities

Para além dos elementos especificados em 4 acima para a descrição do serviço, este serviço inclui ainda a secção FeatureTypeList e algumas das seguintes secções:

I. ServesGMLObjectType – define uma lista de tipos de objectos GML, que não é derivada de gml:AbstractFeatureType e que está disponível no WFS que suporta a operação de GetGMLObject. Estes tipos podem ser definidos no esquema de GML base, ou num esquema de aplicação que usa o seu próprio namespace.
II. SupportsGMLObjectType – define uma lista de tipos de objectos GML que um servidor de WFS deveria de ser capaz de servir.
III. Filter_Capabilities – O esquema desta secção está definido na Filter Encoding Implementation Specification (Vretanos, 2005). Esta é uma secção opcional. Se existir, então o WFS deveria suportar as operações anunciadas. Se a secção não estiver definida, então o cliente deveria assumir que o servidor só suporta os operadores de filtro por defeito, definidos na Filter Encoding Implementation Specification (Vretanos, 2005).

A secção <FeatureTypeList> contém o elemento <Operations> que define as operações comuns a todos os tipos de características e um ou mais elementos <FeatureType> que descrevem cada tipo de característica que o serviço oferece.

A tabela que se segue especifica as operações que são possíveis de definir no elemento <Operations>.

Nome	Descrição
Insert	O elemento <Insert> é usado para indicar que o WFS é capaz de criar novas características.
Update	O elemento <Update> indica que o WFS pode mudar o estado existente de uma característica.

Nome	Descrição
Delete	O elemento <Delete> indica que o WFS pode apagar ou remover características da base de dados.
Query	O elemento <Query> indica que o WFS é capaz de executar uma questão em um tipo de característica.
Lock	O elemento <Lock> indica que o WFS é capaz de obter uma fechadura para uma característica.

Tabela 13 – Lista de acções de transacções possíveis no serviço WFS

As operações de transacção e de query podem ser especificadas globalmente para todos os tipos de característica ou localmente para cada tipo de característica específico contido dentro do elemento <FeatureTypeList>. As operações de transacção e query especificadas globalmente são herdadas por todos os tipos de características contidos dentro do elemento <FeatureTypeList>.

Os elementos seguintes podem ser usados para descrever cada tipo de característica contido dentro do elemento <FeatureTypeList>:

Elemento	Descrição
Name	Nome qualificado para o tipo de característica. Este elemento é obrigatório.
Title	O <Title> é um título humano-legível para identificar este tipo de característica em catálogos.
Abstract	O <Abstract> é uma narrativa descritiva relativa ao tipo de característica.
Keyword	O elemento <Keyword> contém palavras curtas para ajudar nas procuras em catálogos.
DefaultSRS	O elemento <DefaultSRS> indica qual o sistema de referência espacial que será usado pelo WFS para expressar o estado de uma característica espacial caso não tenha sido identificado explicitamente dentro de uma query ou pedido de transacção.
OtherSRS	O elemento <OtherSRS> é usado para indicar outros SRS's suportados dentro do pedido de transacção e query.
NoSRS	O elemento <NoSRS> é usado para tipos de características que não têm nenhuma propriedade espacial, e por isso nenhum SRS. Não é uma exigência para GML que características e FeatureCollections tenham propriedades espaciais. O elemento <NoSRS> nunca implicará, e não pode ser usado para semânticas de "SRS Desconhecido".
Operations	O elemento <Operations> define quais são as operações suportadas por um tipo de característica. Qualquer operação definida localmente tem precedência sobre qualquer operação definida globalmente. Algumas operações podem ser excluídas porque não podem ser suportadas por uma implementação de WFS para um determinado SRS.
OutputFormats	Esta é uma lista de MIME types que indicam os formatos de output que podem ser gerados para um tipo de característica. Se este elemento opcional não é especificado, então são assumidos todos os formatos de output listados para a operação de GetFeature.

Elemento	Descrição
WGS84BoundingBox	O elemento <WGS84BoundingBox> é usado para indicar as extremidades de um rectângulo delimitador em graus decimais de latitude e longitude em WGS84. O seu objectivo é facilitar procuras geográficas indicando onde as instâncias de um tipo de característica particular existem.
MetadataURL	Um WFS pode usar zero ou mais elementos <MetadataURL> para oferecer metadados detalhados sobre os dados de um tipo de característica particular. O atributo *type* indica o padrão usado; o atributo de format indica como os metadados são estruturados. Três tipos padrões podem ser definidos: 'ISO19115' = ISO TC211 19115; 'ISO19139 = ISO TC211 ISO19139; 'FGDC' = FGDC CSDGM.

Tabela 14 – Elemento que descrevem tipos de características

4.2. DescribeFeatureType

Um WFS deve ser capaz perante um pedido de descrever a estrutura de qualquer tipo de característica que pode servir. A descrição é feita usando um esquema, que definem como uma implementação de WFS espera que instâncias de característica sejam codificadas no input (por pedidos Insert e Update) e como as instâncias de características serão geradas no output (em respostas a pedidos GetFeature e GetGmlObject). O único output obrigatório no que diz respeito a um pedido de DescribeFeatureType é o GML3, contudo outros formatos podem ser aceitáveis se anunciados pelo serviço de WFS no documento de capacidades.

Parâmetro	Obrigatório	#	Valor por defeito	Descrição
REQUEST	Sim	1	DescribeFeatureType	Nome da operação
TYPENAME	Não	0+ a)		Uma lista separada por vírgulas de tipos de características a descrever. Se nenhum valor é especificado, então será interpretado como um pedido de todos os tipos de características.
OUTPUTFORMAT	Não	0/1	text/xml; subtype=gml/3.1.1	O formato de produção a usar para descrever os tipos de características. O formato text/xml; subtype=gml/3.1.1 devem ser suportado. Outros formatos de produção são também possíveis se anunciados no documento de capacidades do servidor.
a) Número de elementos na lista				

Tabela 15 - Parâmetros de DescribeFeatureType

Valores	Descrição
XMLSCHEMA	Este valor existe para possibilitar uma compatibilidade com versões anteriores, e é usado para indicar que a resposta ao pedido DescribeFeatureType deve ser gerada usando o esquema GML2.
text/xml; subtype=gml/2.1.2	Este valor é análogo ao XMLSCHEMA
text/xml; subtype=gml/3.1.1	Este valor indica que a resposta ao pedido DescribeFeatureType deve ser gerada usando o esquema GML3. Este é o valor pode defeito se nenhum outro valor for especificado.

Tabela 16 - Valores para o atributo outputFormat

Exemplo:

```
http://www.servidor.com/wfs?SERVICE=WFS&VERSION=1.1.0&REQUEST=DescribeFeatureType&TYPENAME
=TreesA_1M,BuiltUpA_1M&
```

ou

```
<?xml version="1.0" ?>
<DescribeFeatureType
    version="1.1.0"
    service="WFS"
    xmlns="http://www.opengis.net/wfs"
    xmlns:myns="http://www.servidor.com/myns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <TypeName>myns:RoadL_1M</TypeName>
    <TypeName>myns:RoadL_1M</TypeName>
</DescribeFeatureType>

<?xml version="1.0" ?>
<wfs:FeatureCollection
    xmlns="http://www.servidor.com/myns"
    xmlns:myns="http://www.servidor.com/myns"
    xmlns:wfs="http://www.opengis.net/wfs"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs wfs.xsd
```

```
                    http://www.sevidor.com/myns ex.xsd">

    <gml:boundedBy>
        <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
                <gml:lowerCorner>-180.0 -90.0</gml:lowerCorner>
                <gml:upperCorner>180.0 90.0</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>

    <gml:featureMember>
        <RoadL_1M>
                <wkbGeom>
                        <gml:LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
                                <gml:posList>-59.478340 -52.226578 -59.484871 -52.223564</gml:posList>
                        </gml:LineString>
                </wkbGeom>
                <DESIGNATION>HYW 401</DESIGNATION>
                <SURFACE_TYPE>ASPHALT</SURFACE_TYPE>
                <NLANES>12</NLANES>
        </RoadL_1M>
    </gml:featureMember>

</wfs:FeatureCollection>
```

Onde um exemplo de um esquema para este documento poderia ser:

```
<?xml version="1.0" ?>
    <schema
        targetNamespace="http://www.servidor.com/myns"
        xmlns:myns="http://www.servidor.com/myns"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:gml="http://www.opengis.net/gml"
        elementFormDefault="qualified" version="0.1">

    <import namespace="http://www.opengis.net/gml"
    schemaLocation="../gml/3.1.1/base/gml.xsd"/>
    <!-- ============================================
    define global elements
    ============================================= -->

    <element name="RoadL_1M" type="myns:RoadL_1M_Type" substitutionGroup="gml:_Feature"/>
```

```
<!-- ============================================
define complex types (classes)
============================================ -->
<complexType name="RoadL_1M_Type">
    <complexContent>
        <extension base="gml:AbstractFeatureType">
            <sequence>
                <element name="wkbGeom" type="gml:LineStringPropertyType" nillable="false"/>
                <element name="designation" nillable="true" minOccurs="0">
                <simpleType>
                    <restriction base="string">
                        <maxLength value="30"/>
                    </restriction>
                </simpleType>
                </element>
                <element name="surfaceType" nillable="true" minOccurs="0">
                    <simpleType>
                        <restriction base="string">
                            <maxLength value="30"/>
                        </restriction>
                    </simpleType>
                </element>
                <element name="nLANES" nillable="true" minOccurs="0">
                    <simpleType>
                        <restriction base="integer">
                            <totalDigits value="2"/>
                        </restriction>
                    </simpleType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
</schema>
```

4.3. GetFeature

A operação de GetFeature permite obter características de um WFS.

Se um servidor WFS suporta o *Xlink transversal*, então um cliente de WFS pode usar os atributos *traverseXlinkDepth* e *traverseXlinkExpiry* para pedir que propriedades remotas.

O elemento <GetFeature> contém um ou mais elementos <Query>, onde cada destes contém a descrição de uma query. Os resultados de todas as queries contidas em um pedido de GetFeature são concatenados num único output.

O atributo opcional *outputFormat* especifica o formato da resposta a um pedido de GetFeature. O valor por defeito é *text/xml; subtype=gml/3.1.1* que indica que um documento GML3 válido deve ser gerado. Por motivos de compatibilidade documentos de GML2 válidos podem também ser gerados.

O conteúdo do elemento <wfs:FeatureCollection> é controlado pelo valor do atributo *resultType* no elemento <GetFeature>. Se o valor do atributo *resultType* é *results* (valor por defeito), então o WFS tem que gerar uma resposta completa como todo o conteúdo do elemento <wfs:FeatureCollection>. Porém, se é o valor do atributo *resultType* é *hits*, um WFS tem que gerar um elemento <wfs:FeatureCollection> sem conteúdo (i.e. vazio) e actualizado com os respectivos valores dos atributos *timeStamp* e *numberOfFeatures*. Deste modo um cliente pode obter o número de características que uma query pode devolver sem sofrer o custo de transmitir o conjunto inteiro dos resultados.

Para o pedido <GetFeatureWithLock>, um WFS tem que gerar um resultado que inclui o identificador da fechadura. O identificador da fechadura é definido usando o atributo lockId, que está definido no elemento <wfs:FeatureCollection>. O fragmento de XML seguinte ilustra como incluir o atributo lockId na resposta a uma operação:

```
<wfs:FeatureCollection lockId="00A01"... >
...
</wfs:FeatureCollection>
```

A Tabela 17 seguinte apresenta um resumo dos parâmetros que são possíveis no pedido GetFeature e GetFeatureWithLock.

Componente URL	Obrigatório	Valor por defeito	Descrição
REQUEST=[GetFeature | GetFeatureWithLock]	Sim		Nome do pedido WFS
OUTPUTFORMAT	Não	text/xml; subtype=gml/3.1.1	O formato de output a usar na resposta text/xml;subtype=gml/3 .1.1 deve ser suportado. Outros formatos de output são possíveis desde que anunciados no documento de capacidades.
RESULTTYPE	Não	Results	O parâmetro de *resulttype* é usado para indicar se um WFS deveria gerar um documento de resposta completo ou um documento de resposta vazio que indica só o número de características que a query devolveria. Um valor *results* indica que uma resposta completa deveria ser gerada. Um valor *hits* indica que só uma contagem do número de características deveria ser devolvida.

Componente URL	Obrigatório	Valor por defeito	Descrição
PROPERTYNAME	Não		Uma lista de propriedades a obter pode ser especificada para cada tipo de característica no pedido. O carácter "*" pode ser usado para indicar que todas as propriedades deveriam ser obtidas. Há uma correspondência de 1:1 entre cada elemento numa lista FEATUREID ou TYPENAME e a lista de PROPERTYNAME. Na sua ausência todas as propriedades devem ser obtidas.
FEATUREVERSION= [ALL | N]	Não		Se o histórico das características é suportado pelo servidor, o parâmetro FEATUREVERSION orienta o WFS para a versão da característica que é desejada. Caso o valor seja ALL indica que todas as versões de uma característica são desejadas. Se nenhum valor é indicado, então é a última versão da característica que é desejada.
MAXFEATURES=N	Não		É um inteiro positivo que indica o número máximo de características que o WFS deveria devolver como resposta a uma query. Se nenhum valor é especificado então todos os resultados devem ser apresentados.
EXPIRY=N	Não		Este parâmetro só pode ser especificado se o pedido for GetFeatureWithLock. Indica quanto tempo (em minutos) uma fechadura será mantida nas características do conjunto de resultados. Se o parâmetro não é especificado então as fechaduras serão mantidos indefinidamente.
SRSNAME	Não		Este parâmetro é usado para especificar que um determinado SRS suportado pelo WFS deve ser usado nas geometrias das características devolvidas. O valor pode ser o de DefaultSRS ou qualquer outro de OtherSRS que um WFS declara que suporta no documento de capacidades. Se o parâmetro não é especificado, então é usado o valor do DefaultSRS.
TYPENAME	Sim (Apenas se FEATUREID não é especificado)		Uma lista de tipo de característica a questionar.
FEATUREID (Mutuamente exclusivo com FILTER e BBOX)	Não		Uma lista enumerada de características a obter, identificada pelos identificadores de cada característica.
FILTER (Pré-requisito: TYPENAME) (Mutuamente exclusivo com FEATUREID e BBOX)	Não		Descreve o conjunto de características em questão. O filtro tem que estar definido de acordo com Filter Encoding Specification (Vretanos, 2005). Se o parâmetro FILTER é usado, então deve ser especificado um filtro para cada tipo de característica listado no parâmetro TYPENAME.

Componente URL	Obrigatório	Valor por defeito	Descrição
BBOX (Pré-requisito: TYPENAME) (Mutuamente exclusivo com FEATUREID e FILTER)	Não		Em vez de FEATUREID ou FILTER, um cliente pode especificar uma zona delimitadora como descrito na subcláusula 2.
SORTBY	Não		O parâmetro de SORTBY é usado para especificar uma lista de nomes de propriedade que devem de ser usados para ordenar o conjunto de características que será devolvido a uma query. O valor do parâmetro de SORTBY terá a forma "PropertyName [A|D][,PropertyName [A|D],...]" onde a letra A é usada para indicar um tipo ascendente e a letra D é usado para indicar um tipo descendente. Se nem A nem D são especificados, a ordem do tipo em falta será ascendente. Um exemplo concreto poderia ser: "SORTBY=Field1 D, Field2 D, Field3 A" Neste caso os resultados são ordenados primeiro pelo Campo 1, que é ordenado de modo descendente, depois pelo Campo 2 também descendente e por fim pelo Campo 3 ascendente.

Tabela 17 - GetFeature & GetFeatureWithLock encoding

O exemplo que se segue ilustra como questionar algumas propriedades de um conjunto enumerado de características. Neste caso concreto, são obtidos os atributos wkbGeom e tileId da característica "InWaterA_1M.1013", e apenas o atributo wkbGeom da característica "BuiltUpA_1M.3456"

O pedido poderá estar formatado do seguinte modo através do método HTTP GET:

```
http://www.someserver.com/script?
    SERVICE=WFS&
    VERSION=1.1.0&
    REQUEST=GetFeature&
    PROPERTYNAME=InWaterA_1M/wkbGeom,InWaterA_1M/tileId,BuiltUpA_1M/wkbGeom&
    FEATUREID=InWaterA_1M.1013,BuiltUpA_1M.3456
```

Ou do seguinte modo através do método HTTP POST:

```
<?xml version="1.0" ?>
<wfs:GetFeature
    service="WFS"
```

```
    version="1.1.0"
    outputFormat="text/xml; subtype=gml/3.1.1"
    xmlns:myns="http://www.someserver.com/myns"
    xmlns:wfs="http://www.opengis.net/wfs"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <wfs:Query typeName="myns:InWaterA_1M">
        <wfs:PropertyName>myns:wkbGeom</wfs:PropertyName>
        <ogc:Filter>
        <ogc:Not>
        <ogc:Disjoint>
        <ogc:PropertyName>myns:geoTemp</ogc:PropertyName>
        <gml:Envelope srsName="EPSG:4326">
        <gml:lowerCorner>-57.9118 46.2023<gml:lowerCorner>
        <gml:upperCorner>-46.6873 51.8145</gml:upperCorner>
        </gml:Envelope>
        </ogc:Disjoint>
        </ogc:Not>
        </ogc:Filter>
    </wfs:Query>
</wfs:GetFeature>
```

4.4. GetGmlObject

O pedido GetGmlObject providencia uma interface através da qual um WFS pode ser pedido a resolver XLinks para características e elementos que este serve. A operação GetGmlObject é opcional e uma implementação de WFS não precisa de a suportar.

Os parâmetros para este pedido são descritos na Tabela 18.

Parâmetro	Obrigatório	Valor por defeito	Descrição
REQUEST=[GetGmlObject]	Sim		O nome do pedido de WFS.
TRAVERSEXLINKDEPTH	Sim		A profundidade para a qual se tenta resolver uma propriedade XLink. A gama de valores válidos consiste em inteiros positivos mais "*" para o ilimitado.
TRAVERSEXLINKEXPIRY	Não		O número de minutos que um WFS deve esperar para receber uma resposta a um pedido GetGmlObject. Se nenhum valor é especificado então o período é dependente da implementação.

Parâmetro	Obrigatório	Valor por defeito	Descrição
GMLOBJECTID	Sim		O ID do elemento XML a ir buscar.

Tabela 18 - Parâmetros possíveis no pedido GetGmlObject

O exemplo que se segue apresenta um pedido GetGmlObject que vai buscar a característica com o identificador "InWaterA_1M.1234"

```
http://www.servidor.com/wfs?SERVICE=WFS&VERSION=1.1.0&REQUEST=GetGmlObject&TRAVERSEXLINKDEPTH=1&EXPIRY=1&GMLOBJECTID=InWaterA_1M.1234&
```

Este exemplo é semelhante ao anterior mas através do método HTTP POST e definindo o número de XLinks transversais para ilimitado num WFS que tem uma capacidade de GetGmlObject:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetGmlObject
        xmlns:wfs="http://www.opengis.net/wfs"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:gml="http://www.opengis.net/gml"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd"
    service="WFS"
    version="1.1.0"
    outputFormat=" text/xml; subtype=gml/3.1.1"
    traverseXlinkDepth="*"
    traverseXlinkExpiry="2">
    <ogc:GmlObjectId gml:id="t1"/>
</wfs:GetGmlObject>
```

Se o servidor recebe a resposta remota dentro do intervalo de tempo especificado, incorpora-a na resposta a entregar ao cliente:

```
<Town gml:id="t1">
        <gml:name>Bedford</gml:name>
        <gml:directedNode orientation="+">
                <gml:Node gml:id="n1">
                        <gml:pointProperty>
```

```
                    <gml:Point gml:id="townHall" srsName="...">
                        <gml:pos>147 234</gml:pos>
                    </gml:Point>
                </gml:pointProperty>
            </gml:Node>
        </gml:directedNode>
</Town>
```

4.5. Transaction

Um serviço de característica web pode ser capaz de efectuar pedidos de transacção. Um pedido de transacção é composto por operações que modificam as características; ou seja, é composto por operações que inserem, actualizam e removem características geográficas.

A operação *Transaction* é usada para descrever operações de transformação de dados que são aplicadas a características acessíveis pela web. Quando a transacção é completada, o serviço de WFS gera um documento de resposta XML que indica o estado da conclusão da transacção.

A operação de transacção é opcional e uma implementação de WFS não precisa de a suportar para estar conforme a especificação. Contudo se é suportada deve ser anunciada no documento de capacidades.

A única operação de transacção suportada através de HTTP GET é a operação DELETE. As outras operações como INSERT e UPDATE devem ser efectuadas através de HTTP POST, pois geralmente tornam-se bastante longas quando codificadas através de pares chave-valor.

4.5.1. DELETE

O elemento <DELETE> é usado para indicar uma ou mais instâncias de características que devem ser apagadas. A operação de apagar é restrita as características abrangidas pelo elemento <FILTER> como descrito em Filter Encoding Implementation Specification. No caso em que o elemento <FILTER> não identifica nenhuma instância de característica a apagar, a acção apagar não terá qualquer efeito. Esta não é uma condição de excepção.

O exemplo seguinte elimina todas as características de tipo InWaterA_1M e BuiltUpA_1M que estão dentro do filtro especificado usando o método HTTP GET.

```
http://www.servidor.com/script?
SERVICE=WFS&
VERSION=1.1.0&
REQUEST=Transaction&
OPERATION=Delete&
TYPENAME=InWaterA_1M,BuiltUpA_1M&
FILTER=(<Filter><Within><PropertyName>InWaterA_1M/wkbGeom<PropertyName>
```

```
<gml:Envelope>
<gml:lowerCorner>10 10</gml:lowerCorner>
<gml:upperCorner>20 20</gml:upperCorner></gml:Envelope></Within>
</Filter>)(< Filter><Within><PropertyName>BuiltUpA_1M/wkbGeom
<PropertyName><gml:Envelope><gml:lowerCorner>10 10</gml:lowerCorner>
<gml:upperCorner>20,20</gml:upperCorner>
</gml:Envelope></Within></Filter>)
```

O exemplo seguinte elimina características usando o método HTTP POST.

```
<?xml version="1.0" ?>
<wfs:Transaction
    version="1.1.0"
    service="WFS"
    xmlns="http://www.servidor.com/myns"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:wfs="http://www.opengis.net/wfs"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <wfs:Delete typeName="InWaterA_1M">
    <ogc:Filter>
    <ogc:GmlObjectId gml:id="InWaterA_1M.1013"/>
    </ogc:Filter>
    </wfs:Delete>
</wfs:Transaction>
```

4.5.2. INSERT

O elemento <INSERT> é usado para criar novas características. Por defeito, o estado inicial de uma característica a ser criada é expresso usando GML3 e tem que ser valido relativamente ao esquema de GML3 gerado pela operação DescribeFeatureType. Porém, o atributo de *inputFormat* pode ser definido para suportar outras versões mais antigas de GML.

Múltiplos elementos <Insert> podem ser incluídos num único pedido *Transaction* e múltiplas características podem ser criadas com um único elemento <Insert>.

O atributo idgen definido no elemento <Insert> pode ser usado para indicar que método deve ser usado para atribuir identificadores às características.

Tabela 19 define os valores possíveis para o atributo, assim como o seu significado.

Valor para Idgen	Acção
GenerateNew (por defeito)	O WFS gerará identificadores únicos para todas as características inseridas.
UseExisting	Na resposta a uma operação <Insert>, um WFS usará os valores do atributo gml:id nas características e elementos inseridos. Se algum destes ID's estiver duplicado relativamente a um já guardado, o WFS responderá com uma excepção.
ReplaceDuplicate	Um cliente WFS pode pedir ao serviço para gerar um ID, de modo a substituir os inputs de atributos gml:id de características que estejam duplicados em vez de apresentar uma excepção.

Tabela 19 - Valores para o atributo idegen

A transacção seguinte cria duas instâncias do tipo de característica InWaterA_1M.

```
<?xml version="1.0"?>
<wfs:Transaction
    version="1.1.0"
    service="WFS"
    xmlns="http://www.servidor.com/myns"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:wfs="http://www.opengis.net/wfs"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www. servidor.com/myns
                        http://www. servidor.com/wfs/wfs?request=describefeaturetype&amp;typename=InWaterA_1M.xsd
                        http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <wfs:Insert idgen="UseExisting">
        <InWaterA_1M gml:id="INW1">
            <wkbGeom>
                <gml:Polygon gml:id="P1" srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
                    <gml:exterior>
                        <gml:LinearRing>
                            <gml:posList>-98.54 24.26 ...</gml:posList>
                        </gml:LinearRing>
                    </gml:exterior>
                </gml:Polygon>
            </wkbGeom>
            <id>150</id>
            <f_code>ABCDE</f_code>
            <hyc>152</hyc>
```

```
                <tileId>250</tileId>
                <facId>111</facId>
        </InWaterA_1M>
        <InWaterA_1M gml:id="INW2">
                <wkbGeom>
                        <gml:Polygon gml:id="P2" srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
                                <gml:exterior>
                                        <gml:LinearRing>
                                                <gml:posList>-99.99 22.22 ...</gml:posList>
                                        </gml:LinearRing>
                                </gml:exterior>
                        </gml:Polygon>
                </wkbGeom>
                <id>111</id>
                <f_code>FGHIJ</f_code>
                <hyc>222</hyc>
                <tileId>333</tileId>
                <facId>444</facId>
        </InWaterA_1M>
    </wfs:Insert>
</wfs:Transaction>
```

É assumido que o esquema de referência para InWaterA_1M.xsd seja criado usando a operação *DescribeFeatureType*. Neste exemplo, o documento é estaticamente referenciado, mas pode ser dinamicamente referenciado.

4.5.3. UPDATE

O elemento <Update> descreve uma operação de actualização que será aplicada a uma característica ou conjunto de características de um único tipo de característica. Múltiplas operações <Update> podem ser descritas em um único pedido *Transaction*.

Um elemento <Update> contém um ou mais elementos <Property> que especificam o nome (<Name>) e um valor opcional de substituição (<Value>) para uma propriedade que pertence ao tipo de característica especificado através do atributo obrigatório *typeName*. A omissão do elemento <Value> significa que a propriedade deve ser atribuída a um valor NULL. No caso de a propriedade não poder ter valores nulos, um WFS terá que responder com uma excepção que indica que o valor NULO não é permitido.

O exemplo seguinte actualiza a propriedade "population" da característica identificada pelo identificador de característica BuiltUpA_1M.1013.

```
<?xml version="1.0" ?>
<wfs:Transaction
    version="1.1.0"
    service="WFS"
    xmlns="http://www.servidor.com/myns"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:wfs="http://www.opengis.net/wfs"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <wfs:Update typeName="BuiltUpA_1M">
        <wfs:Property>
            <wfs:Name>population</wfs:Name>
            <wfs:Value>4070000</wfs:Value>
        </wfs:Property>
        <ogc:Filter>
            <ogc:GmlObjectId gml:id="BuiltUpA_1M.10131"/>
        </ogc:Filter>
    </wfs:Update>
</wfs:Transaction>
```

4.6. LockFeature

Devido ao facto do protocolo HTTP ser stateless, as semânticas de transacções não são preservadas. De modo a entender o assunto, considere uma operação de actualização.

O cliente obtém uma característica do servidor. A característica é modificada pelo cliente e submetida de volta para a base de dados por um pedido de Transacção para actualização. É perdida a serialização porque não há nada que garanta que enquanto a característica estava a ser modificada pelo cliente, não houve outro cliente que actualizou a mesma característica na base de dados.

Um modo para assegurar a serialização é requerer que o acesso a dados seja feito de uma maneira mutuamente exclusiva; isto é, enquanto uma transacção estiver a aceder aos dados, nenhuma outra transacção os pode modificar. Isto pode ser realizado usando fechaduras.

O objectivo desta operação é expor um mecanismo de fechadura a longo termo sobre uma característica de modo a assegurar a sua consistência.

Parâmetro	Obrigatório	Valor por defeito	Descrição
REQUEST	Sim	LockFeature	
EXPIRY	Não		Numero de minutos que a fechadura deve existir. Se não for especificado nenhum valor então esta durará indefinidamente

Parâmetro	Obrigatório	Valor por defeito	Descrição
LOCKACTION=[ALL | SOME]	Não		Especifica o modo com que as fechaduras são adquiridas. ALL indica que tem que obter uma fechadura para todas as características, caso contrário falha. SOME, tenta obter uma fechadura para o maior número possível de características
TYPENAME	Sim (Apenas se FEATUREID não é especificado)		Nome de um ou mais tipos de características que se pretendem bloquear.
FEATUREID (Mutuamente exclusivo com FILTER e BBOX)	Não		Lista que enumera os identificadores das características que se pretendem bloquear.
FILTER (Pré-requisito: TYPENAME) (Mutuamente exclusivo com FEATUREID e BBOX)	Não		Descreve o conjunto de características em questão. O filtro tem que estar definido de acordo com Filter Encoding Specification (Vretanos, 2005). Se o parâmetro FILTER é usado, então deve ser especificado um filtro para cada tipo de característica listado no parâmetro TYPENAME.
BBOX (Pré-requisito: TYPENAME) (Mutuamente exclusivo com FEATUREID e FILTER)	Não		Em vez de FEATUREID ou FILTER, um cliente pode especificar uma zona delimitadora como descrito na subcláusula 2.

Tabela 20 – Operação LockFeature

O exemplo seguinte bloqueia todas as instâncias de características do tipo InWaterA_1M e BuiltUpA_1M.

```
http://www.servidor.com/script?
    SERVICE=WFS&
    VERSION=1.1.0&
    REQUEST=LockFeature&
    TYPENAME=InWaterA_1M,BuiltUpA_1M&
```

Ou

```
    <?xml version="1.0" ?>
    <LockFeature
```

```
    version="1.1.0"
    service="WFS"
    xmlns="http://www.opengis.net/wfs"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:myns="http://www. servidor.com/myns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <Lock typeName="myns:InWaterA_1M"/>
</LockFeature>
```

Podendo receber como resposta:

```
<?xml version="1.0" ?>
<WFS_LockFeatureResponse
    xmlns="http://www.opengis.net/wfs"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <LockId> A10143758D< /LockId>
</WFS_LockFeatureResponse>
```

Um exemplo onde são especificadas as características pode ser:

```
<?xml version="1.0" ?>
<LockFeature
    version="1.1.0"
    service="WFS"
    lockAction="SOME"
    xmlns="http://www.opengis.net/wfs"
    xmlns:myns="http://www.servidor.com/myns"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <Lock typeName="myns:InWaterA_1M">
        <ogc:Filter>
            <ogc:GmlObjectId gml:id="InWaterA_1M.1013"/>
            <ogc:GmlObjectId gml:id="InWaterA_1M.1014"/>
            <ogc:GmlObjectId gml:id="InWaterA_1M.1015"/>
            <ogc:GmlObjectId gml:id="InWaterA_1M.1016"/>
            <ogc:GmlObjectId gml:id="InWaterA_1M.1017"/>
        </ogc:Filter>
```

```
    </Lock>
</LockFeature>
```

Uma possível resposta ao pedido poderia ser:

```
<?xml version="1.0" ?>
<WFS_LockFeatureResponse
    xmlns="http://www.opengis.net/wfs"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <LockId>1</LockId>
    <FeaturesLocked>
        <ogc:GmlObjectId gml:id="InWaterA_1M.1013"/>
        <ogc:GmlObjectId gml:id="InWaterA_1M.1014"/>
        <ogc:GmlObjectId gml:id="InWaterA_1M.1016"/>
        <ogc:GmlObjectId gml:id="InWaterA_1M.1017"/>
    </FeaturesLocked>
    <FeaturesNotLocked>
        <ogc:GmlObjectId gml:id="InWaterA_1M.1015"/>
    </FeaturesNotLocked>
```

Outro exemplo de um pedido que usa combinações poderia ser:

```
<LockFeature
    version="1.1.0"
    service="WFS"
    expiry="4"
    lockAction="SOME"
    xmlns="http://www.opengis.net/wfs"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:myns="http://www.servidor.com/myns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <Lock handle="LOCK1" typeName="myns:BuiltUpA_1M">
            <ogc:Filter>
            <ogc:Within>
```

```
                <ogc:PropertyName>BuiltUpA_1M/wkbGeom</ogc:PropertyName>
                <gml:Polygon gid="1"
                srsName="http://www.opengis.net/gml/epsg.xml#63266405">
                <gml:exterior>
                <gml:LinearRing>
                <gml:posList>-95.7 38.1 -97.8 38.2 ...</gml:posList>
                </gml:LinearRing>
                </gml:exterior>
                </gml:Polygon>
                </ogc:Within>
                </ogc:Filter>
        </Lock>
        <Lock handle="LOCK2" typeName="myns:InWaterA_1M">
                <ogc:Filter>
                <ogc:GmlObjectId gml:id="InWaterA_1M.1212"/>
                <ogc:GmlObjectId gml:id="InWaterA_1M.1213"/>
                <ogc:GmlObjectId gml:id="InWaterA_1M.10"/>
                </ogc:Filter>
        </Lock>
        </LockFeature>
```

Por fim, de modo a ilustrar um exemplo de uma resposta a uma transacção que actualiza, elimina e insere, é apresentado o seguinte exemplo:

```
<?xml version="1.0" ?>
    <wfs:TransactionResponse
    version="1.1.0"
    xmlns:wfs="http://www.opengis.net/wfs"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
    <wfs:TransactionSummary>
        <wfs:totalInserted>2</wfs:totalInserted>
        <wfs:totalUpdated>4</wfs:totalUpdated>
        <wfs:totalDeleted>2</wfs:totalDeleted>
    </wfs:TransactionSummary>
    <wfs:TransactionResults>
       <wfs:Action locator="STMT2" code="9999">
       <wfs:Message>Insert Action Failed</wfs:Message>
    </wfs:Action>
    </wfs:TransactionResults>
```

```
    <wfs:InsertResults>
        <wfs:Feature handle="STMT1">
        <ogc:FeatureId fid="SomeFeature.4567"/>
      </wfs:Feature>
        <wfs:Feature handle="STMT1">
        <ogc:FeatureId fid="SomeFeature.4568"/>
      </wfs:Feature>
        <wfs:Feature handle="STMT1">
        <ogc:FeatureId fid="SomeFeature.4569"/>
      </wfs:Feature>
        <wfs:Feature handle="STMT3">
        <ogc:FeatureId fid="Feature2.389345"/>
      </wfs:Feature>
    </wfs:InsertResults>
</wfs:TransactionResponse>
```

Anexo IV. Web Processing Service (WPS)

"WPS define uma interface padrão que facilita a publicação de processos geoespaciais, e a descoberta e uso destes por clientes" (Open Geospatial Consortium, 2008).

Um WPS pode ser configurado para oferecer qualquer tipo de funcionalidade GIS a clientes através da rede, incluindo o acesso a cálculos pré-programados e/ou modelos de computação que operam em dados georreferenciados. Um WPS pode oferecer cálculos simples como subtrair um conjunto de números georreferenciados por outro conjunto análogo (por exemplo, determinar a diferença de casos de gripe entre duas estações diferentes), ou tão complicado quanto um modelo de mudança de clima global assim como pode ter um numero qualquer para entradas e saídas de dados. Os dados requeridos pelo WPS podem ser transmitidos pela rede ou estarem disponíveis no servidor.

O WPS pode ser usado não só para processamento de dados numéricos como também para o processamento de dados vectoriais, raster ou outro tipo de dados.

Neste contexto Publicação (Publishing), significa tornar disponível informação máquina-legível assim como também metadados humano-legível, que permite a descoberta e uso de serviços.

1. Operações OGC WPS

A interface de WPS especifica três operações com muitas semelhanças as dos outros Serviços Web do OGC já referidos (WMS, WFS e OWS), que podem ser pedidas por um cliente e serem executadas por um servidor de WPS.

Todas as operações são obrigatórias em todos os servidores. As operações exigidas são as seguintes:

- GetCapabilities – Operação que permite a um cliente pedir e receber documentos com metadados do serviço, que descrevem as capacidades da implementação específica do servidor. Esta operação também suporta negociação da versão de especificação a ser usada para interacções de cliente-servidor (Open Geospatial Consortium, 2008).
  A resposta a um pedido de GetCapabilities é um documento de XML que contém metadados sobre o serviço, assim como, uma breve descrição de todos os processos implementados.

- DescribeProcess – Esta operação permite para um cliente receber informação detalhada sobre os processos suportados pelo serviço, incluído parâmetros de entrada necessários, formatos permitidos, e parâmetros de saída que podem ser produzidos (Open Geospatial Consortium, 2008).

- Execute – Esta operação permite a um cliente correr um processo específico implementado pelo WPS, usando os parâmetros de entrada fornecidos e devolvendo os outputs produzidos (Open Geospatial Consortium, 2008).

1.1. GetCapabilities

A Tabela 21 contém todos os argumentos possíveis num pedido de GetCapabilities assim a sua multiplicidade.

Parâmetro	Multiplicidade	Obrigatório	Valor por defeito	Descrição
SERVICE	1	Sim	WPS	Nome do serviço
REQUEST	1	Sim	GetCapabilities	Nome da operação
AcceptVersions	0/1	Não		Versões que o cliente aceita

Tabela 21 - Parâmetros possíveis num pedido WPS GetCapabilities

Exemplo de um pedido de GetCapabilities limitado à versão 0.1.0:

```
http://servidor:porto/script?SERVICE=WPS&REQUEST=GetCapabilities&AcceptVersions=0.1.0&
```

O documento de resposta ao pedido GetCapabilities deve conter as secções ServiceIdentification, ServiceProvider e OperationsMetadata que apresentam poucas diferenças relativamente aos esquemas da OWS e também o elemento ProcessOfferings contendo uma lista não ordenada de descrições sobre os processos (ProcessBrief) oferecidos no servidor de WPS. Deve ainda incluir o parâmetro obrigatório "version" e o parâmetro opcional updateSequence.

Tabela 22 contém os elementos que podem fazer parte do elemento ProcessBrief.

Elemento	Multiplicidade	Obrigatório	Domínio	Definição
Identifier	1	Sim	ows:CodeType	Identificador inequívoco para um processo, input ou output.
Title	1	Sim	String	Título de um processo, input ou output, normalmente disponível para apresentar ao utilizador.
Abstract	0/1	Não	String	Descrição de um processo, input ou output, normalmente disponível para apresentação ao utilizador.
Metadata	0/1	Não	ows:Metadata	Referência para mais metadados sobre este processo. É incluído quando se acha que é útil.
processVersion	0/1	Não	ows:VersionType	Versão do processo.

Tabela 22 - Elementos possíveis em ProcessBrief

O exemplo para o elemento ProcessOfferings onde um Process é declarado no esquema de XML como sendo do tipo ProcessBrief.

```
<ProcessOfferings>
        <Process processVersion="1.0">
                <ows:Identifier>buffer</ows:Identifier>
                <ows:Title>Buffer a polygon feature</ows:Title>
                <ows:Abstract>
                 Buffer the polygon coordinates found in one GML stream by a given buffer distance, and output the results in GML.
                </ows:Abstract>
                <ows:Metadata xlink:title="buffer" />
                <ows:Metadata xlink:title="polygon" />
        </Process>
</ProcessOfferings>
```

1.2. DescribeProcess

A Tabela 23 contém um sumário de todos os parâmetros que podem ser incluídos na operação DescribeProcess.

Parâmetro	Obrigatório	Multiplicidade	Valor por defeito	Definição
SERVICE	Sim	1	WPS	Nome do serviço
REQUEST	Sim	1	DescribeProcess	Nome da operação
VERSION	Sim	1		Versão usada na comunicação
IDENTIFIER	Sim	1+		Identificador do processo

Tabela 23 - Possíveis parâmetros para a operação DescribeProcess

Um exemplo de um pedido DescribeProcess usando HTTP GET poderia ser:

```
http://www.servidor.com/script?Service="WPS"&Request="DescribeProcess"&Version=0.1.0&Identifier=intersection,buffer&
```

E um exemplo do mesmo pedido usando HTTP POST poderia ser:

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

```
<DescribeProcess service="WPS" version="0.1.0"
        xmlns="http://www.opengeospatial.net/wps" xmlns:ows="http://www.opengeospatial.net/ows"
        xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengeospatial.net/wps
        ..\wpsDescribeProcess.xsd">
        <ows:Identifier>intersection</ows:Identifier>
        <ows:Identifier>buffer</ows:Identifier>
</DescribeProcess>
```

A resposta esperada de um pedido DescribeProcess válido é uma estrutura de dados ProcessDescriptons que contém as descrições dos processos correspondentes aos identificadores incluídos no pedido. Cada processo pode ter qualquer número de parâmetros de entrada ou saída.

Cada parâmetro é descrito por uma estrutura de dados que especifica os formatos admissíveis, codificações e unidades de medida (quando aplicável). Para cada parâmetro de entrada, o processo pode especificar que necessita de um dos seguintes elementos: ComplexData, LiteralData ou BoundingBoxData (ver detalhes mais a frente).

ProcessDescriptons é um elemento que contém uma lista de ProcessDescripton

Elemento	#	Obrigatório	Domínio	Descrição
Identifier	1	Sim	ows:CodeType	Ver Tabela 22 acima
Title	1	Sim	String	
Abstract	0/1	Não	String	
Metadata	1	Sim	ows:Metadata	
processVersion	0/1	Não	ows:VersionType	
ProcessInputs	0/1	Não	ProcessInputs	Lista dos parâmetros de entrada exigidos e opcionais para o processo. Geralmente é requerido pelo menos um.
ProcessOutputs	1	Sim	ProcessOutputs	Lista de parâmetros de saída exigidos da execução deste processo.
storeSupported	0/1	Não	Booleano (true ou false) [1]	Indica se os parâmetros de saída de dados complexos deste processo podem ser armazenados pelo servidor de WPS, podendo ser posteriormente acessíveis por um URL. Pode ser útil quando o processo demora muito tempo a processar. O valor por defeito é false.
statusSupported	0/1	Não	Booleano (true ou false) [1]	É recomendado quando o processo demora muito tempo a executar. E significa que a qualquer momento um cliente pode questionar o estado do pedido.

Elemento	#	Obrigatório	Domínio	Descrição
1) Por defeito (false) a resposta ao pedido Execute é recebida quando o pedido termina o processamento.				

Tabela 24 - Elementos possíveis em ProcessDescription

O elemento ProcessInputs é constituído por uma lista de elementos ProcessInput que estão descritos na Tabela 25.

Elemento	#	Obrigatório	Domínio	Valor por defeito	Descrição
Identifier	Ver Tabela 24				
Title					
Abstract					
MinimumOccurs	0/1	Não	Inteiro (0 ou 1)	1	Identifica se o parâmetro é opcional ou não.
InputFormChoice	1	Sim	InputFormChoice		Identifica o tipo de dados de entrada e fornece informação de suporte.

Tabela 25 - Elementos possíveis em InputDescription

Elemento	#	Obrigatório	Tipo	Descrição
ComplexData	0/1	Condicional[1]	Supported-ComplexData	Indica que o parâmetro de entrada será uma estrutura de dados complexa (como p.e um fragmento de GML), e fornece uma lista de formatos, codificações e esquemas suportados[2]
LiteralData	0/1	Condicional[1]	LiteralInput	Indica que o parâmetro de entrada será um valor literal simples (como um numero) que está embutido no pedido Execute e descreve os valores possíveis
BoundingBoxData	0/1	Condicional[1]	SupportedCRSs	Indica que o parâmetro de entrada será uma estrutura de dados BoundingBox que está embutida no pedido Execute. Fornece também uma lista com os CRSs suportados
1) Apenas um dos três itens deve ser incluído 2) O valor desta estrutura de dados complexa pode ser um parâmetro de entrada embutido dentro do pedido Execute ou remotamente acessível ao servidor				

Tabela 26 - Elementos possíveis em InputFormChoise

ProcessOutputs são listas de elementos ProcessOutput descritos na Tabela 27.

Atributo	#	Obrigatório	Domínio	Definição
Identifier	Ver Tabela 24			
Title				
Abstract				
OutputFormChoice	1	Sim	OutputFormChoice	Identifica o tipo deste parâmetro de saída e fornece informação de suporte

Tabela 27 - Elementos possíveis em OutputDescription

Elemento	#	Obrigatório	Domínio	Definição
ComplexOutput	0/1	Condicional[1]	Supported-ComplexData	Indica que este parâmetro de saída será um conjunto de dados complexo (como um fragmento de GML), e fornece listas de formatos, codificações e esquemas suportados[2]
LiteralOutput	0/1	Condicional[1]	LiteralOutput	Indica que este parâmetro de saída será um valor literal simples (como um inteiro) que está embutido na resposta de execução e identifica os valores possíveis
BoundingBoxOutput	0/1	Condicional[1]	SupportedCRSs	Indica que este parâmetro de saída será uma estrutura de dados BoundingBox que está embutida na resposta da execução e fornece uma lista do CRSs suportados nestas Bounding Boxes

1) Apenas um dos três itens deve ser incluído
2) O cliente pode assim seleccionar de entre os formatos, codificações e esquemas identificados. O valor desta estrutura de dados complexa pode ser embutido dentro da resposta de execução ou estar remotamente acessível ao cliente.

Tabela 28 - Elementos possíveis em OutputFormChoise

Elemento	#	Obrigatório	Domínio	Definição
SupportedComplexData	0+	Não	SupportedComplexData	Combinações de formatos, codificações, e/ou esquemas suportado pelo parâmetro de entrada ou de saída do processo[1]

Elemento	#	Obrigatório	Domínio	Definição
defaultFormat	0/1	Não	ows:MimeType	Informação sobre qual é o formato por defeito para o parâmetro de entrada ou de saída do processo[2]
defaultEncoding	0/1	Não	URI	Informação sobre qual é a codificação por defeito para o parâmetro de entrada ou de saída do processo[3]
defaultSchema	0/1	Não	URI	Informação sobre qual é o esquema XML por defeito para o parâmetro de entrada ou de saída do processo[4]

1) Esta estrutura de dados deveria ser incluída quando o processo suporta mais de uma combinação de formato / codificação / esquema XML e será usada para cada combinação suportada. Esta estrutura de dados não será incluída se existir apenas uma única combinação.
2) Este parâmetro será incluído quando o formato por defeito é diferente de text/xml. É opcional caso contrário.
3) Este parâmetro será incluído quando a codificação por defeito é diferente da codificação XML no documento resposta (por exemplo. UTF-8).
4) Este parâmetro será omisso quando não existir nenhum esquema de XML associado ao parâmetro (por exemplo um arquivo de GIF). Este parâmetro será incluído quando o parâmetro é codificado em XML usando um esquema de XML. Quando incluído, o parâmetro deverá respeitar o esquema de XML referido.

Tabela 29 - Elementos possíveis em ComplexData

Elemento	#	Obrigatório	Tipo	Definição
Format	0+	Não	ows:MimeType	Identificação do formato suportado pelo parâmetro de entrada ou de saída do processo[1]
Encoding	0+	Não	URI	Referência à codificação suportada pelo parâmetro de entrada ou de saída do processo[2]
Schema	0+	Não	URI	Referência a esquemas XML suportados pelo parâmetro de entrada ou de saída do processo[3]

1) Este elemento será incluído se o formato for diferente do especificado em defaultFormat.
2) Este elemento será incluído se a codificação for diferente da especificada em defaultEncoding.
3) Este elemento será incluído se o esquema for diferente do especificado em defaultSchema. Cada um destes elementos de XML ou tipos será definido em um documento de esquema XML separado.

Estes elementos não serão incluídos se houver apenas um único valor ou quando este não se aplicar.

Tabela 30 - Elementos possíveis em SupportedComplexData

Elemento	Multiplicidade	Obrigatório	Domínio	Definição
CRS	1+	Sim	URI	Referência para um CRS
defaultCRS	1	Sim	URI	Referência para um CRS por defeito

Tabela 31 - Elementos possíveis em SupportedCRSs

Elemento	#	Obrigatório	Domínio	Definição
DataType	0/1	Não	DomainMetadata	Tipo de dados para este parâmetro
SupportedUOMs	0/1	Não	SupportedUOMs	Lista de unidades de medida suportadas por este parâmetro numérico

Tabela 32 - Elementos possíveis em LiteralOutput

Elemento	#	Obrigatório	Domínio	Definição
UOM	1+	Sim	DomainMetadata	Unidade de medida. Incluir para cada UOM suportado
defaultUOM	0/1	Não	URI	Referência a uma unidade de medida por defeito, quando esta existe

Tabela 33 - Elementos possíveis em SupportedUOMs

Elemento	#	Obrigatório	Tipo	Definição
DataType	Ver Tabela 32			
SupportedUOMs				
LiteralValuesChoice	1	Sim	LiteralValuesChoice	Identifica tipo de literal e fornece informação de suporte
DefaultValue	0/1	Não	CharacterString	Valor por defeito, caso este exista

Tabela 34 - Elementos possíveis em LiteralInput

Elemento	#	Obrigatório	Tipo	Definição
AllowedValues	0/1	Condicional[1]	AllowedValues	Indica que existe um conjunto finito de valores e intervalos permitidos por este parâmetro e apresenta uma lista ordenada de todos os valores e/ou intervalos válidos
AnyValue	0/1	Condicional[1]	DomainMetadata	Indica que qualquer valor é permitido para este parâmetro
ValuesReference	0/1	Condicional[1]	ValuesReference	Indica que há um conjunto finito de valores e intervalos permitido por este parâmetro, especificados na lista referenciada
1) Apenas um dos três itens deve ser incluído				

Tabela 35 - Elementos possíveis em LiteralValuesChoise

Uma resposta à operação DescribeProcess para WPS pode ter o seguinte aspecto codificado em XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps"
        xmlns:wps="http://www.opengeospatial.net/wps" xmlns:ows="http://www.opengeospatial.net/ows"
        xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengeospatial.net/wps
        ..\wpsDescribeProcess.xsd">
        <ProcessDescription processVersion="2"
                storeSupported="true" statusSupported="false">
        <ows:Identifier>Buffer</ows:Identifier>
                <ows:Title>Create a buffer around a polygon.</ows:Title>
                <ows:Abstract>Create a buffer around a single polygon. Accepts the polygon as GML and provides GML
                output for the buffered feature. </ows:Abstract>
                <ows:Metadata xlink:title="spatial" />
                <ows:Metadata xlink:title="geometry" />
                <ows:Metadata xlink:title="buffer" />
                <ows:Metadata xlink:title="GML" />
                <DataInputs>
                        <Input>
                                <ows:Identifier>InputPolygon</ows:Identifier>
                                <ows:Title>Polygon to be buffered</ows:Title>
                                <ows:Abstract>URI to a set of GML that describes the polygon.</ows:Abstract>
                                <ComplexData defaultFormat="text/XML" defaultEncoding="base64"
                                        defaultSchema="http://www.servidor.com/gml/3.1.0/polygon.xsd">
                                        <SupportedComplexData>
                                                <Format>text/XML</Format>
                                                <Encoding>UTF-8</Encoding>
                                                <Schema>http://server.com/gml/3.1.0/polygon.xsd</Schema>
                                        </SupportedComplexData>
                                </ComplexData>
                                <MinimumOccurs>1</MinimumOccurs>
                        </Input>
                        <Input>
                                <ows:Identifier>BufferDistance</ows:Identifier>
                                <ows:Title>Buffer Distance</ows:Title>
                                <ows:Abstract>URI to a GML resource file</ows:Abstract>
                                <LiteralData>
                                        <SupportedUOMs defaultUOM="meters"/>
                                        <ows:AnyValue/>
                                </LiteralData>
                                <MinimumOccurs>1</MinimumOccurs>
                        </Input>
```

```
            </DataInputs>
            <ProcessOutputs>
                <Output>
                    <ows:Identifier>BufferedPolygon</ows:Identifier>
                    <ows:Title>Buffered Polygon</ows:Title>
                    <ows:Abstract>GML stream describing the buffered polygon feature.</ows:Abstract>
                    <ComplexOutput defaultFormat="text/XML" defaultEncoding="base64"
                        defaultSchema="http://www.servidor.com/gml/3.1.0/polygon.xsd">
                        <SupportedComplexData>
                            <Format>text/XML</Format>
                            <Encoding>UTF-8</Encoding>
                            <Schema>http://servidor.com/gml/3.1.0/polygon.xsd</Schema>
                        </SupportedComplexData>
                    </ComplexOutput>
                </Output>
            </ProcessOutputs>
        </ProcessDescription>
</ProcessDescriptions>
```

1.3. Execute

Os ouputs podem ser devolvidos como uma resposta directa ao pedido. Alternativamente, o servidor pode armazenar o resultado como recursos acessíveis web. Se os resultados são armazenados, a resposta à operação consistirá num documento de XML que inclui um URL para cada parâmetro de saída armazenado que o cliente pode usar para obter o resultado.

A resposta pode ser devolvida depois de o processamento terminar, ou alternativamente, pode ser devolvida imediatamente a seguir à aceitação pelo servidor. Nesse último caso, a resposta inclui apenas a informação sobre o estado do processo, que indica se o processo completou ou não, como também um URL de estado. O URL de estado devolve uma resposta actualizada sobre o estado do processo e permite ao cliente saber também o tempo restante para que este complete.

Elemento	#	Obrigatório	Valor por defeito	Domínio	Definição
SERVICE	1	Sim	WPS	String	Nome do serviço
REQUEST	1	Sim	Execute	String	Nome da operação
VERSION	1	Sim		String	Versão usada na comunicação
IDENTIFIER	1+	Sim		ows:CodeType	Identificação do processo

Elemento	#	Obrigatório	Valor por defeito	Domínio	Definição
DataInputs	0/1	Não		DataInputs	Lista de parâmetros de entrada passados[1] DataInputs contém uma lista não ordenada de IOValue, onde IOValue é constituído pelos elementos Identifier, Title, Abstract (opcional) e ValueFormChoice
OutputDefinitions	0/1	Não		OutputDefinitions	Lista de definições de parâmetros de saída desejados da execução deste processo[2]
Store	0/1	Não	False	Booleano (true/false)	Especifica se todos os parâmetros de saída complexos deste processo devem ser armazenados como recursos web-acessíveis[3]
Status	0/1	Não	False	Booleano (true/false)	Especifica se a resposta a Execute será devolvida após a aceitação[4]

1) É possível não exista nenhum parâmetro de entrada só quando todos os parâmetros de entrada são pré-determinados recursos fixos. Em todos os outros casos, é requerido pelo menos um.
2) Estes parâmetros de saída não são normalmente identificados, a menos que o cliente peça um subconjunto limitado destes, e/ou peça formatos, esquemas e/ou codificações diferente das que estão definidas por defeito.
3) Se "store" for "true", o servidor armazenará todos os resultados complexos do processo online para que o cliente os possa obtê-los como exigido. Se "store" for "false", todo os resultados complexos serão codificados dentro da resposta à operação Execute.
   O parâmetro "store" não deve ser incluído a menos que o parâmetro "storeSupported" seja "true" e esteja incluído no ProcessDescription para este processo. O valor de parâmetro "store", "true" é recomendado para fazer encadeamento de serviços mais eficiente quando o resultado for de grandes dimensões.
4) Se o parâmetro "status" for "true", o servidor deve devolver a resposta do Execute assim que for aceite a execução deste processo. Se "status" for "falso", o servidor não devolverá a resposta do Execute até que a execução do processo esteja completa.
   Este parâmetro de "status" não será incluído a menos que o parâmetro de "statusSupported" seja "true" e esteja incluído no ProcessDescription para este processo. O valor de parâmetro de "status", "true", é recomendado quando um processo pode levar muito tempo para executar.

Tabela 36 - Elementos possíveis em Execute

Elemento	#	Obrigatório	Tipo	Descrição
ComplexValueReference	0/1	Condicional[1]	ValueReference	Identifica o valor deste parâmetro como um recurso web acessível e referência esse recurso[2]
ComplexValue	0/1	Condicional[1]	ComplexValue	Identifica o valor do parâmetro como uma estrutura de dados complexa e fornece o valor[3]

Elemento	#	Obrigatório	Tipo	Descrição
LiteralValue	0/1	Condicional[1]	LiteralValue	Identifica o valor do parâmetro como um valor literal e fornece esse valor
BoundingBoxValue	0/1	Condicional[1]	ows:BoundingBox	Identifica o valor do parâmetro como uma estrutura de dados BoundingBox e fornece o valor

1) Apenas um dos três itens deve ser incluído
2) Para um parâmetro de entrada, esta estrutura de dados pode ser usada pelo cliente quando codificado como ComplexData no ProcessDescription. Para um parâmetro de saída, este elemento será usado por um servidor quando "store" no pedido Execute é "true"
3) Para um parâmetro de entrada, esta estrutura de dados pode ser usada pelo cliente quando codificado como ComplexData no ProcessDescription. Para um parâmetro de saída, este elemento será usado por um servidor quando quer "storeSupported" no ProcessDescription ou "store" dentro do pedido Execute é "true"

Tabela 37 - Elementos possíveis em ValueFormChoice

Elemento	#	Obrigatório	Domínio	Definição
format	Ver ComplexValueEncoding			
encoding				
schema				
reference	1	Sim	URL	Referência para recurso acessível na Web e usado como parâmetro de entrada, ou fornecido por um processo como parâmetro de saída

Tabela 38 - Elementos possíveis em ValueReference

Elemento	#	Obrigatório	Domínio	Definição
format	Ver ComplexValueEncoding			
encoding				
schema				
reference	1	Sim	Qualquer	Valor complexo a ser usado como parâmetro de entrada a processar

Tabela 39 - Elementos possíveis em ComplexValue

Elemento	#	Obrigatório	Domínio	Descrição
value	1	Sim	String	Valor

Elemento	#	Obrigatório	Domínio	Descrição
dataType	0/1	Não	URI	Identificador do tipo de dados deste valor literal
uom	0/1	Não	URI	Identificador da unidade de medida deste valor numérico literal

Tabela 40 - Elementos possíveis em LiteralValue

O Output é uma lista de elementos OutputDefinition descritos abaixo.

Elemento	#	Obrigatório	Tipo	Descrição
Identifier	Ver Tabela 24			
Title				
Abstract				
format	Ver ComplexValueEncoding			
encoding				
schema				
uom	0/1	Não	URI	Identificador da unidade de medida pedida para este parâmetro de saída

Tabela 41 - Elementos possíveis em OutputDefinition

Exemplo de um pedido Execute:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Execute service="WPS" version="0.4.0" store="true" status="false"
        xmlns="http://www.opengeospatial.net/wps"
        xmlns:ows="http://www.opengeospatial.net/ows"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
        <ows:Identifier>Buffer</ows:Identifier>
        <DataInputs>
                <Input>
                        <ows:Identifier>InputPolygon</ows:Identifier>
                        <ows:Title>Playground area</ows:Title>
                        <ComplexValueReference    ows:reference="http://www.server.com/some_WFS_request.xml"
```

```
                    schema="http://www.server.com/gml_polygon_schema.xsd" />
            </Input>
            <Input>
                    <ows:Identifier>BufferDistance</ows:Identifier>
                    <ows:Title>Distance which people will walk to get to a playground</ows:Title>
                    <LiteralValue uom="meters">400</LiteralValue>
            </Input>
        </DataInputs>
        <OutputDefinitions>
            <Output>
                    <ows:Identifier>BufferedPolygon</ows:Identifier>
                    <ows:Title>Area serviced by playground.</ows:Title>
                    <ows:Abstract>Area within which most users of this playground will live.</ows:Abstract>
            </Output>
        </OutputDefinitions>
</Execute>
```

Exemplo de um pedido Execute via HTTP GET:

```
http://www.servidor.com/script?
    REQUEST=Execute&
    SERVICE=WPS&
    VERSION=0.4.0&
    store=true&
    Identifier="Buffer"&
   DataInputs=Buffer,http://www.servidor.com/script2,BufferDistance,100
```

Em todos os outros casos, a resposta para um pedido válido de operação Execute é um documento XML com ExecuteResponse. O conteúdo deste ExecuteResponse depende do valor do parâmetro de "store", do número de resultados produzidos e o tipo desses resultados. Se o armazenamento não foi pedido, o documento de ExecuteResponse conterá todos os resultados inline. Se o armazenamento é pedido, o ExecuteResponse vai fazer referência ao recurso web-acessível onde estes podem ser obtidos (Open Geospatial Consortium, 2008).

Nome	#	Obrigatório	Domínio	Definição
version	1	Sim	String	Versão para a especificação
Identifier	1	Sim	ows:CodeType	Identificador ou nome único de um processo
DataInputs	0/1	Não	DataInputs	Lista de parâmetros de entrada

Nome	#	Obrigatório	Domínio	Definição
OutputDefinitions	0/1	Não	OutputDefinitions	Lista de definições de parâmetros de saída desejados do executar deste processo
ProcessOutputs	0/1	Não	ProcessOutputs	Lista de valores de parâmetros de saída da execução do processo
Status	1	Sim	Status	Estado de execução deste processo
statusLocation	0/1	Não	URL	Referência para a localização onde o documento de ExecuteResponse actual está armazenado

Tabela 42 - Elementos possíveis em ExecuteResponse

Elemento	#	Obrigatório	Domínio	Definição
Output	1	Sim	IOValue	Valor do parâmetro de saída

Tabela 43 - Elementos possíveis em ProcessOutputs

Elemento	#	Obrigatório	Domínio	Definição
ProcessAccepted	0/1	Condicional[1]	String	Indica que o processo foi aceite pelo servidor, mas está em fila e ainda não começou a executar.[2]
ProcessStarted	0/1	Condicional[1]	ProcessStarted	Indica que o processo foi aceite pelo servidor e o processamento foi iniciado.
ProcessSucceeded	0/1	Condicional[1]	String	Indica que o processo completou e a execução não teve problemas.[3]
ProcessFailed	0/1	Condicional[1]	ProcessFailed	Indica que a execução deste processo falhou e inclui informação sobre os erros.

4) Apenas um dos quatro itens deve ser incluído
5) A mensagem deve poder ser lida por humanos e deve conter informações sobre o tamanho da fila assim como outras informações possíveis
6) A mensagem deve poder ser lida por humanos se apresentada pelo cliente e pode conter informações úteis, tais como quanto tempo o processo demorou a executar ou problemas que possam ter sido encontrados.

Tabela 44 - Elementos possíveis em Status

Elemento	#	Obrigatório	Domínio	Definição
message	1	Sim	String	Texto humano-legível onde se espera que inclua qualquer mensagens que o servidor possa desejar dar aos clientes
percentCompleted	0/1	Não	Inteiro de 0 a 100 inclusive	Percentagem do processo que foi completada, onde 0 significa que o acabou de começar, e 100 que o processo terminou. É esperado que seja preciso pelo menos dentro do intervalo de 10%

Tabela 45 - Elementos possíveis em ProcessStarted

Elemento	#	Obrigatório	Domínio	Definição
ExceptionReport	1	Sim	ExceptionReport	Relatório de excepção que contém uma ou mais estruturas de dados de Excepção, cada uma identificando um erro diferente

Tabela 46 - Elementos possíveis em ProcessFailed

Quando um processo termina com sucesso, a estrutura de dados de *Status* incluirá o parâmetro de ProcessSucceeded e a estrutura de dados ProcessOutputs será preenchida completamente.

No caso em que os Outputs sejam armazenados como recursos web, estes são identificados por um elemento "reference", como mostrado no fragmento de XML:

<ComplexResultReference xlink:href="http://www.servidor.com/caminho/resultados.xml"/>

Também a informação disponível no URL de estado será actualizada com um documento de ExecuteResponse actualizado. Se o URL incluído em um ComplexValueReference é acedido antes de o processo ter tempo de completar o servidor devolverá um Erro 403 (Not Found).

O documento de ExecuteResponse normalmente contém os parâmetros de entrada que foram fornecidos pelo cliente, nas estruturas de dados DataInputs e OutputDefinitions. Esta informação inclui ainda qualquer URIs fornecidos dentro do pedido de execução. Se os parâmetros de entrada estavam embutidos no pedido, então o servidor pode gerar e fornecer um URL para visualização e fazer referência a ela, em vez de devolver uma cópia do original no corpo do ExecuteResponse. Isto é recomendado no caso de arquivos grandes.

A resposta no exemplo seguinte ilustra o uso do elemento statusLocation. Este atributo contém um URL para aceder a um documento de ExecuteResponse com a última informação sobre o estado da execução do processo. Se o processo terminou de executar o URL fará referência à resposta. Se o processo ainda não completou, a localização da resposta não será necessariamente identificada. Neste exemplo, porque o pedido de execução especificou status = "false", o URL à qual a informação de estado fica situada será povoada quando o processo completou. Quer isto dizer, que o WPS apenas entra a resposta quando o processo termina, e o URL terá um conteúdo idêntico ao da resposta de execução.

```
<?xml version="1.0" encoding="UTF-8"?>
<ExecuteResponse statusLocation="http://www.server.com/execute_response_uri.xml" version="0.4.0"
        xmlns="http://www.opengeospatial.net/wps" xmlns:ows="http://www.opengeospatial.net/ows"
        xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
        instance" xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
        <ows:Identifier>Buffer</ows:Identifier>
        <Status>
                <ProcessSucceeded/>
        </Status>
        <DataInputs>
                <Input>
                        <ows:Identifier>InputPolygon</ows:Identifier>
                        <ows:Title>Playground area</ows:Title>
                        <ComplexValueReference ows:reference="http://www.server.com/some_WFS_request.xml"
                        schema="http://www.server.com/gml_polygon_schema.xsd" />
                </Input>
                <Input>
                        <ows:Identifier>BufferDistance</ows:Identifier>
                        <ows:Title>Distance which people will walk to get to a playground</ows:Title>
                        <LiteralValue uom="meters">400</LiteralValue>
                </Input>
        </DataInputs>
        <OutputDefinitions>
                <Output>
                        <ows:Identifier>BufferedPolygon</ows:Identifier>
                        <ows:Title>Area serviced by playground.</ows:Title>
                        <ows:Abstract>Area within which most users of this playground will live.</ows:Abstract>
                </Output>
        </OutputDefinitions>
        <ProcessOutputs>
                <Output>
                        <ows:Identifier>BufferedPolygon</ows:Identifier>
                        <ows:Title>Area serviced by playground.</ows:Title>
                        <ows:Abstract>
                                Area within which most users of this playground will live
                        </ows:Abstract>
                        <ComplexValueReference ows:reference="http://www.server.com/buffered_polygon.xml"/>
                </Output>
        </ProcessOutputs>
</ExecuteResponse>
```

2. Excepções

Quando um servidor de WPS encontra um erro enquanto executa uma operação Execute, deve devolver uma mensagem de excepção. Os códigos de excepção padrão estão listados abaixo.

Código de Excepção	Significado do código	Valor do parâmetro "locator"
MissingParameterValue	O pedido não inclui um valor de parâmetro e este servidor não declarou um valor por defeito para aquele parâmetro	Nome do parâmetro em falta
InvalidParameterValue	O pedido contém um valor de parâmetro inválido	Nome do parâmetro com valor inválido
NoApplicableCode	Nenhum outro exceptionCode especificado por este serviço e servidor se aplica a esta excepção	Nenhum (o parâmetro não é aplicado)
ServerBusy	O servidor está muito ocupado para aceitar e fazer fila para o pedido neste momento	Nenhum (o parâmetro não é aplicado)
FileSizeExceeded	O tamanho do arquivo de um dos parâmetros de input era muito grande	Nenhum (o parâmetro não é aplicado)

Tabela 47 - Código de Excepção para WPS

Anexo V. Styled Layer Descriptor e Symbology Encoding

Styled Layer Descriptor (SLD) é um esquema de XML especificado pelo Open Geospatial Consortium que estende a especificação do Web Map Server de modo a permitir uma personalização da aparência de uma característica e dos dados de cobertura. A habilidade para definir estas simbolizações definidas pelo utilizador requer uma linguagem de personalização que tanto o cliente como o servidor possam entender. Esta linguagem é chamada Symbology Encoding (SE).

Se os mapas necessitarem de ser gerados dinamicamente tendo em conta muitos factores que descrevem o actual utilizador, situação e contexto, é possível então dar uso a esta especificação que se pode tornar numa poderosa ferramenta descritiva.

Anexo VI. Esquema para uma Excepção XML OGC

Este anexo contém o esquema de XML para excepções de serviço.

Uma excepção de XML é validada de acordo com o esquema da excepção do serviço. Num ambiente HTTP, o MIME type devolvido é "text/xml". As mensagens de erro individuais aparecem como elementos <ServiceException> dentro do elemento <ServiceExceptionReport>. As mensagens podem ser formatadas como pedaços densos de plain text ou incluídas numa secção de dados (CDATA).

O atributo "*locator*" é um texto opcional que um servidor pode usar para indicar que parte de um pedido de serviço causou a excepção a ser gerada. É pretendido que o locator seja interpretado por humanos em vez de através de software de cliente, e como tal não tem nenhuma sintaxe predefinida.

O esquema que se segue é o esquema por defeito do OGC

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="http://www.opengis.net/ogc"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xs:element name="ServiceExceptionReport">
    <xs:annotation>
      <xs:documentation> The ServiceExceptionReport element contains one or more
        ServiceException elements that describe a service exception. </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ServiceException" type="ogc:ServiceExceptionType" minOccurs="0"
          maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation> The Service exception element is used to describe a
              service exception. </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="version" type="xs:string" fixed="1.2.0"/>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="ServiceExceptionType">
    <xs:annotation>
      <xs:documentation> The ServiceExceptionType type defines the ServiceException element.
        The content of the element is an exception message that the service wished to convey
        to the client application. </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="code" type="xs:string">
          <xs:annotation>
            <xs:documentation> A service may associate a code with an exception by using
              the code attribute. </xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="locator" type="xs:string" use="optional">
```

```
            <xs:annotation>
              <xs:documentation> The locator attribute may be used by a service to
                 indicate to a client where in the client's request an exception was
                 encountered. If the request included a 'handle' attribute, this may be
                 used to identify the offending component of the request. Otherwise the
                 service may try to use other means to locate the exception such as line
                 numbers or byte offset from the begining of the request, etc ...
              </xs:documentation>
            </xs:annotation>
          </xs:attribute>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

Anexo VII. Exemplo de GetCapabilities de um serviço WMS OGC

Ver ficheiro WMS_GetCapabilities.xml existente na pasta "Anexos\Documentos" no CD em anexo.

Anexo VIII. Exemplo de GetCapabilities de um serviço WFS OGC

Ver ficheiro WFS_GetCapabilities.xml existente na pasta "Anexos\Documentos" no CD em anexo.

Anexo IX. Exemplo de GetCapabilities do Serviço WPS OGC

Ver ficheiro WPS_GetCapabilities.xml existente na pasta "Anexos\Documentos" no CD em anexo.

Anexo X. Questionário

Devido às dimensões do ficheiro é possível apenas visualiza-lo na pasta "Anexos\Documentos" no CD em anexo.

Anexo XI. Vídeos

No CD em anexo é possível encontrar os seguintes vídeos:

- Interface WFS-T.avi – Vídeo que ilustra o funcionamento da interface WFS-T
- Interface WMS.avi – Vídeo que ilustra como os mapas podem ser visualizados, adicionados e modificados (estilo)
- Interface WPS.avi – Vídeo que ilustra o funcionamento da interface WPS
- Visualização de Dados Georreferenciados.avi – Vídeo que ilustra como se pode beneficiar da visualização de dados, e usar a sua referencia geográfica de modo mais eficaz

Bibliografia

3D-Stadtmodell Berlin. (2008). Obtido em 6 de Janeiro de 2009, de http://www.3d-stadtmodell-berlin.de

Andrienko, N., & Andrienko, G. (2004). Interactive visual tools to explore spatio-temporal variation. *AVI'04: Proceedings of the working conference on Advanced visual interfaces.*

Beaujardiere, J. d. (Março de 2006). OpenGIS Implementation Specification #06-042: OpenGIS Web Map Server Implementation Specification.

Berlin 3D. (2008). Obtido em 27 de Agosto de 2008, de 3D RealityMaps: http://www.realitymaps.de

Berners-Lee, T., Fielding, R., Fielding, R., & Masinter, L. (Agosto de 1998). *IETF RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax*. Obtido em 14 de Novembro de 2008, de The Internet Engineering Task Force: http://www.ietf.org/rfc/rfc2396.txt

Borges, J. L. (1998). Obras completas I. Rio de Janeiro: Globo.

Boucelma, O., Lacroix, Z., & Essid, M. (2002). *A wfs-based mediation system for gis interoperability.* NY, USA: ACM Press.

Bray, T., Hollander, D., Layman, A., & Tobin, R. (16 de Agosto de 2006). *Namespaces in XML*. Obtido em 14 de Dezembro de 2008, de W3C Recommendation: http://www.w3.org/TR/REC-xml-names/

Bray, T., Paoli, J., & Sperberg-McQueen, C. M. (Outubro de 2000). Obtido em 4 de Dezembro de 2008, de Extensible Markup Language (XML) 1.0: http://www.w3.org/TR/1998/REC-xml-19980210

Brooks, S., & Whalley, J. (2008). *Multilayer hybrid visualizations to support 3D GIS.* Elsevier.

Brooks, S., & Whalley, L. (2005). A 2D/3D hybrid geographical information system. *Proc. of the 3rd intern. conf. on Computer graphics and interactive techniques in Australasia and South East Asia.* ACM.

Brown, J., & McGregor, A. (2000). *Network performance visualization: Insight through animation.* In Proc. of PAM2000.

Card, S., Mackinlay, J., & Shneiderman, B. (1999). *Readings in Information Visualization.* San Francisco, CA: Morgan Kaufmann Publishers.

Card, S., Mackinlay, J., & Shneiderman, B. (1999). *Readings in Information Visualization.* San Francisco, CA: Morgan Kaufmann Publishers.

CEN. (2008). *The european committee for standardization*. Obtido em 27 de Abril de 2009, de http://www.cen.eu

Chopra, A. (2004). *Programming with XMLBeans.*

Clark, J., & DeRose, S. (16 de Novembro de 1999). *XML Path Language (XPATH), Version 1.0*. Obtido em 14 de Novembro de 2008, de W3C Recommendation: http://www.w3.org/TR/xpath.html

Clark, R. W., & Maurer, R. (2006). *Visual terrain editor: an interactive editor for real terrains.* J. Comput. Small Coll. 22 (2).

Collins, C., & Carpendale, S. (2007). *VisLink: Revealing relationships amongst visualizations.* IEEE Transactions on Visualization and Computer Graphics (Proceedings of the IEEE Conference on Information Visualization.

Conselleria d'Infraestructures i Transport. (2008). *gvSIG.* Obtido em 5 de Dezembro de 2008, de http://www.gvsig.gva.es

Conti, G., & De Amicis, R. (2008). New generation 3d web-based geographical information systems - The importance of integrated infrastructures for territory management. *In proceedings of Webist.* Madeira, Portugal.

Conti, G., Andreolli, M., Piffer, S., & De Amicis, R. (2008). A 3D web based geographical information system for regional planning. *In proceedings of Webist.* Madeira, Portugal.

Conti, G., De Amicis, R., & Simões, B. (2009). A 3D interface for orchestration of web process for environmental control. *IEEE VR* .

Conti, G., De Amicis, R., Piffer, S., & Simões, B. (2008). Multi-level service infrastructure for geovisual analytics in the context of territorial management. *IJITSA* .

Coughlin, J., Cuff, S., & Krause, N. (2008). SERVIR Viz: A 3D data access and visualization system for mesoamerica. *Directions Magazione Worldwide Source for Geospatial Technology* .

CRS. (07 de Julho de 2006). *Information and Service System for European Coordinate Reference Systems.* Obtido em 11 de Novembro de 2008, de http://crs.bkg.bund.de/crs-eu/

Darken, R., & Sibert, J. (1993). A Toolset for Navigation in Virtual Environments. *Proceedings of the UIST '93* , 157-160.

Darken, R., & Sibert, J. (1996). Navigating in Large Virtual Worlds. *The International Journal of Human-Computer Interaction* , 49-72.

De Amicis, R., Conti, G., Simões, B., Lattuca, R., Tosi, N., Piffer, S., et al. (2008). Geo-Visual Analytics for Land Planning and Urban Design. *Proceedings of IDMME - Virtual Concept 2008* .

De Amicis, R., Witzel, M., & Conti, G. (2007). Interoperable networked service-based infrastructure for interactive. *In Proceedings of Proceedings of the NATO-OTAN Workshop on Development of a Prototype System for Sharing Information related to Acts of Terrorism to the Environment, Agriculture and Water systems.* Venice.

DeeGree. (2009). *Free Software for Spatial Data Infrastructures.* Obtido em 6 de Janeiro de 2009, de http://www.deegree.org

DeRose, S. J., Daniel, R., Maler, E., & Marsh, J. (25 de Março de 2003). *XPointer xmlns() Scheme.* Obtido em 16 de Outubro de 2008, de W3C Recommendation: http://www.w3.org/TR/xptr-xmlns/

DeRose, S., Maler, E., & Orchard, D. (27 de Junho de 2001). *XML Linking Language (XLink) Version 1.0.* Obtido em 15 de Outubro de 2008, de W3C Recommendation 27 June 2001: http://www.w3.org/TR/xlink/

DG Joint Research Centre, European Commission. (2006). *Report of international workshop on spatial data infrastructures, cost-benefit / return on investment.* Italy: Ispra.

Eccles, R., Kapler, T., Harper, R., & Wright, W. (2008). Stories in GeoTime. *Palgrave Macmillan* .

Eick, S., & Wills, G. (1995). *High Interaction Graphics.* European Journal of Operational Research.

Einsfeld, K., Ebert, A., & Wölle, J. (2007). *Interconnected Media for Human-Centered Understanding.*

Encarnação, J., De Amicis, R., Conti, G., & J., R. (2008). Beyond FOSS 3D gis technologies: a chance for developing countries. *In proceedings of FOSS4G 2008 - Free and Open Source Software for Geospatial Conference.* Cape Town, South Africa.

Environment System Research Institute. (2008). Obtido em 6 de Janeiro de 2009, de http://www.esri.com

EPSG. (12 de Novembro de 2008). *European Petroleum Survey Group Geodesy Parameters*. Obtido em 6 de Janeiro de 2009, de http://www.epsg.org/

ERDAS. (2008). Obtido em 6 de Janeiro de 2009, de Imagine virtual GIS: http://www.erdas.com

ESDIS. (2008). *The earth science data and information system project*. Obtido em 2 de Abril de 2009, de http://esdis.eosdis.nasa.gov

ESRI. (1998). Obtido em 11 de Novembro de 2008, de ESRI Shapefile Technical Description: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

ESRI. (2008). Obtido em 11 de Novembro de 2008, de ArcSDE: http://www.esri.com/software/arcgis/arcsde/

ESRI. (2008). Obtido em 11 de Novembro de 2008, de The GIS Software Leader: http://www.esri.com

ESRI. (June de 2003). *Coming to Terms With Interoperability and Standards*. Obtido em 6 de Janeiro de 2009, de ESRI GIS and Mapping Software: http://www.esri.com/news/arcuser/0403/glossary.html

Fallman, S. (2004). *Using WFS to build GIS support.*

Fielding et. al. (Junho de 1999). *IETF RFC 2616: Hypertext Transfer Protocol – HTTP/1.1*. Obtido em 18 de Novembro de 2008, de http://www.ietf.org/rfc/rfc2616.txt

Freed, N., & Borenstein, N. (Novembro de 1996). *IETF RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. Obtido em 17 de Outubro de 2008, de The Internet Engineering Task Force: http://www.ietf.org/rfc/rfc2045.txt

FreeGIS.org. (2008). *FreeGIS.org*. Obtido em 11 de Novembro de 2008, de http://freegis.org/search?q=Community%20Positioning%20System

FTK - Research Institute for Telecommunication. (2008, May 30). European ICT manifesto for the regions - Identifying future ICT trends of exceptional importance for regional economic change and successfully incorporating them into proactive regional policy approaches. Dortmund.

Furnas, G. (1997). Effective View Navigation. *Proceedings of CHI '97* , 367-374.

Ganesan, D., Estrin, D., & Heidemann, J. (2003). Dimensions: Why do we need a new data handling architecture for sensor networks? *SIGCOMM Comput. Commun.* , pp. 143–148.

GDAL. (2008). *GDAL - Biblioteca de Abstracoes de Dados Geo-Espaciais*. Obtido em 7 de Janeiro de 2009, de http://www.gdal.org

GDI3D. (2008). *Spatial data infrastructure for 3d-geodata*. Obtido em 12 de Janeiro de 2009, de http://www.geographie.uni-bonn.de/karto/hd3d/webstart.en.htm

GEO World. (Agosto de 2003). Canada goes online continued: Ogc web services explode geoprocessing.

GEOS. (2008). Obtido em 13 de Janeiro de 2009, de http://trac.osgeo.org/geos/

GeoServer. (2008). Obtido em 13 de Janeiro de 2009, de http://geoserver.org

GeoTools. (2008). *The Open Source Java GIS Toolkit*. Obtido em 13 de Janeiro de 2009, de Codehaus Foundation: http://geotools.codehaus.org

GeoTools. (July de 2008). *The Open Source Java GIS Toolkit*. Obtido em 22 de Fevereiro de 2009, de Codehaus Foundation: http://geotools.codehaus.org/

Gershon, N., & Eick, S. (1997). *Information visualization applications in the real world.* IEEE.

GISCA. (2008). *Adelaide Model*. Obtido em 13 de Janeiro de 2009, de http://www.gisca.adelaide.edu.au

Global Earth Observation System of Systems. (2008). Obtido em 13 de Janeiro de 2009, de http://www.epa.gov/geoss

Global Monitoring for Environment and Security. (2008). Obtido em 13 de Janeiro de 2009, de http://www.gmes.info

GML4J. (2001). Obtido em 11 de Novembro de 2008, de http://gml4j.sourceforge.net

GMT. (2008). Obtido em 13 de Janeiro de 2009, de http://gmt.soest.hawaii.edu

Google Earth. (2008). Obtido em 13 de Janeiro de 2009, de http://earth.google.com

Gore, A. (1998). *The Digital Earth: understanding our planet in the 21st Century.* Los Angeles: California Science Center.

GRASS GIS. (2008). Obtido em 14 de Janeiro de 2009, de http://grass.osgeo.org/

Grasso, V., & Singh, A. (2008). *Advances in space research.* Elsevier.

Grasso, V., Cervone, G., Singh, A., & Kafatos, M. (2006). Global environmental alert service. *American Geophysical Union, Fall Meeting* .

Haist, J., & Coords, B. (2005). The W3DS-Interface of CityServer3D. *European Spatial Data Research (EuroSDR). Next Generation 3D City Models* , 63-67.

Haist, J., & Korte, P. (2006). Adaptive streaming of 3d gis geometries and textures for interactive visualisation of 3d city models. *9th AGILE Intern. conf. on Geographic Information Science.*

Harries, K. (1999). *Mapping Crime: Principles and Practice.* U.S.: Department of Justice, Office of Justice Programs.

Herring, J. R. (Outubro de 2006). OpenGIS Implementation Specification #06-104: Implementation Specification for Geographic information - Simple feature access.

Hershberger, J., & Snoeyink, J. (1994). An O(n log n) implementation of the Douglas-Peucker algorithm for line simplification. *In Proceedings of the tenth annual symposium on Computational geometry.* New York, United States: ACM Press.

Hetzler, E. G., & Turner, A. (2004). Analysis experiences using information visualization. *IEEE Computer Graphics and Applications, 24(5)* , pp. 22-26.

Huang, B., & Lin, H. (1999). GeoVR: A web-based tool for virtual reality presentation from 2D data. *Computers and Geosciences* , 1167–1175.

Hunt, E., & Waller, D. (1999). *Orientation and wayfinding: A Review ONR Technical Report N00014-96-0380.*

IMPROVE. (s.d.). *IMPROVE.* Obtido em 11 de Janeiro de 2009, de http://www.improve-eu.info

INSPIRE. (2008). Obtido em 11 de Janeiro de 2009, de http://inspire.jrc.ec.europa.eu

Intergraph. (2007). Obtido em 11 de Janeiro de 2009, de Intergraph, GeoMedia Terrain: http://www.intergraph.com

Intergraph SG&I - Geospatially Powered Solution. (2008). Obtido em 14 de Janeiro de 2009, de http://www.intergraph.com/sgi/default.aspx

ISO 19111. (2003). *Geographic information — Spatial referencing by coordinates.*

ISO 8601. (2004). *Data elements and interchange formats — Information interchange — Representation of dates and times.*

ISO. (2008). *International standards for business, government and society*. Obtido em 20 de Abril de 2009, de ISO - International organization for standardization: http://www.iso.org

ISOMETRICS. (2006). Obtido em 21 de Janeiro de 2009, de http://www.isometrics.uni-osnabrueck.de/index.htm

JBoss. (2008). *jboss.org: community driven*. Obtido em 21 de Janeiro de 2009, de www.jboss.org

JOGL. (2008). Obtido em 2 de Maio de 2009, de Java binding for the opengl api: https://jogl.dev.java.net

JTS Topology Suite. (2008). Obtido em 21 de Janeiro de 2009, de http://www.tsusiatsoftware.net/jts/main.html

Kapler, T., & Wright, W. (2004). *GeoTime Information Visualization.* IEEE InfoVis.

Keim, D. (2005). *Presentation at workshop on visual analytics.*

Kersting, O., & Doellner, J. (2002). Interactive 3d visualization of vector data in gis. *Prooc. of the 10th ACM intern. symp. on Advances in geographic information systems* (pp. 107–112). ACM.

Koller, D. et al. (1995). Virtual gis: A real-time 3d geographic information system. *Proc. of the 6th conf. on Visualization* , pp. 94 – 9.

Köller, D., Lindstrom, P., Ribarsky, W., Hodges, L. F., Faust, N., & Turner, G. (1995). Virtual GIS: A real-time 3D geographic information system. *In Proceedings of the IEEE conference on visualization'95* , 94–100.

Kötter, T. (2007). Prevention of Environmental Disasters by Spatial Planning and Land Management.

Kwok, J., & Nithianandan, A. (2006). Visualization of GML Data in OpenMap.

Liarokapis, F., Raper, J., & Brujic-Okretic, V. (2006). Navigating within the urban environment using Location and Orientation-based Services. *European Navigation Conference.* Manchester, UK.

MacEachren, M. (2005). *Moving geovisualization toward support for group work. Exploring geovisualization.* Elsevier.

Manoharan, T., Taylor, H., & Gardiner, P. (2002). A collaborative analysis tool for visualisation and interaction with spatial data. *Proc.of the 7th intern. Conf. on 3D Web technology*, (pp. 75–83).

Mansourian, A., Rajabifard, A., & Zoej, J. (2005). *Development of a web-based gis using sdi for disaster management.* Springer.

Mapbuilder. (2009). Obtido em 1 de Maio de 2009, de http://communitymapbuilder.osgeo.org

MapGuide. (2008). Obtido em 10 de Maio de 2009, de http://mapguide.osgeo.org

MapServer. (2008). Obtido em 27 de Novembro de 2008, de http://mapserver.org

Microsoft Virtual Earth. (2008). Obtido em 25 de Fevereiro de 2009, de http://www.microsoft.com/VirtualEarth

Moura, A. C. (2000). Contribuições Metodológicas do Geoprocessamento à Geografia.

NASA World Wind. (2008). *NASA World Wind*. Obtido em 25 de Fevereiro de 2009, de http://worldwind.arc.nasa.gov

National Visualization and Analytics Center. (2004). *Illuminating the Path: Research and Development Agenda for Visual Analytics* .

Nebert, D., Whiteside, A., & Vretanos, P. (Fevereiro de 2007). OpenGIS Implementation Specification #07-005: OpenGIS Catalogue Services Specification.

Nourbakhsh, I., Sargent, R., Wright, A., Cramer, K., McClendon, B., & Jones, M. (2006). *Mapping disaster zones* (Vol. 439). Nature.

NVAC. (2008). *National visualization and analytics center*. Obtido em 1 de Dezembro de 2008, de http://nvac.pnl.gov

Ohigashi, M., Guo, Z. S., & Tanaka, Y. (2006). Integration of a 2d legacy gis, legacy simulations, and legacy databases into a 3d geographic simulation. *Proc. of the 24th annual conf. on Design of communication* (pp. 149–156). ACM.

Open Geospatial Consortium. (2008). Obtido em 10 de Fevereiro de 2009, de http://www.opengeospatial.org

Open Geospatial Consortium. (2009). Obtido em 12 de Dezembro de 2008, de Compliance and Interoperability Testing Initiative (CITE): http://cite.opengeospatial.org/

Open Geospatial Consortium. (2009). Obtido em 10 de Janeiro de 2009, de Web Service Common - 06-121r3: http://www.opengeospatial.org/standards/common

Open Geospatial Consortium. (12 de Agosto de 2008). *About OGC.* Obtido em 12 de Fevereiro de 2009, de Open Geospatial Consortium: http://www.opengeospatial.org/ogc

Open Geospatial Consortium. (2009). *Abstract Specifications.* Obtido em 30 de Janeiro de 2009, de http://www.opengeospatial.org/standards/as

Open Geospatial Consortium. (2008). *All Registered Products.* Obtido em 10 de Fevereiro de 2009, de Open Geospatial Consortium: http://www.opengis.org/resources/?page=products

Open Geospatial Consortium. (s.d.). *GML - the Geography Markup Language.* Obtido em 10 de Fevereiro de 2009, de http://www.opengis.net/gml/

Open Geospatial Consortium. (2009). *OpenGIS Simple Features for SQL.* Obtido em 27 de Janeiro de 2009, de http://www.opengeospatial.org/standards/sfs

Open Geospatial Consortium. (2009). *Web Coverage Service.* Obtido em 10 de Novembro de 2008, de http://www.opengeospatial.org/standards/wcs

Open Geospatial Consortium. (Agosto de 2008). *Web Feature Service.* Obtido em 11 de Janeiro de 2009, de Open Geospatial Consortium: http://www.opengeospatial.org/standards/wfs

Open Geospatial Consortium. (August de 2008). *Web Map Service.* Obtido em 25 de Fevereiro de 2009, de Open Geospatial Consortium: http://www.opengeospatial.org/standards/wms

Open Geospatial Consortium. (2008). *Web Processing Service.* Obtido em 25 de Fevereiro de 2009, de Open Geospatial Consortium: http://www.opengeospatial.org/standards/wps

OpenEV. (2006). Obtido em 10 de Janeiro de 2009, de http://openev.sourceforge.net

OpenLayers. (2008). Obtido em 20 de Agosto de 2008, de OpenLayers: Free Maps for the Web: http://openlayers.org

Oracle. (2009). Obtido em 10 de Janeiro de 2009, de Oracle Spatial: http://www.oracle.com

Ordnance Survey. (September de 2006). *Geographic information strategy.* Obtido em 13 de Janeiro de 2009, de http://www.ordnancesurvey.co.uk/oswebsite/aboutus/reports/gistrategy/gistrategy.pdf

OSSIM. (2008). Obtido em 22 de Janeiro de 2009, de http://www.ossim.org

Panagopoulos, T. (12 de Julho de 1999). *Glossário dos SIG.* Obtido em 20 de Outubro de 2008, de http://w3.ualg.pt/~tpanago/glossario.htm

PCIGEOMATICS. (2008). Obtido em 22 de Janeiro de 2009, de PAMAP GIS topographer: http://www.pcigeomatics.com

Percivall, G. (2002). *The OpenGIS Abstract Specification, Topic 12: OpenGIS Service Architecture, Version 4.3.*

Plaisant, C., Milash, B., Rose, A., Widoff, S., & Shneiderman, B. (1996). *LifeLines: Visualizing Personal Histories.* ACM.

Portele, C. (2007). *OpenGIS Implementation Specification #07-036: OpenGIS Geography Markup Language (GML) Implementation Specification, version 3.2.1.*

PostGIS. (2001). Obtido em 20 de Abril de 2009, de http://postgis.refractions.net/

PostgreSQL. (1996). Obtido em 1 de Maio de 2009, de http://www.postgres.org

Rajabifard, A., & Williamson, I. (2004). *SDI development and capacity building*. Retrieved Abril 10, 2009, from http://www.geom.unimelb.edu.au/research/SDI_research/

Ramsey, P. (2005). *The State of Open Source GIS.* Obtido em 21 de Abril de 2009, de http://www.bostongis.com/?content_name=oss_gis_state

Reddy, M., Leclerc, Y., Iverson, L., & Bletter, N. (1999). TerraVisionII: Visualizing massive terrain databases in VRML. *IEEE Computer Graphics and Applications* , 30–38.

Refractions Research. (2004). *The State of Open Source GIS.*

Rescorla, E., & Schiffman, A. (Agosto de 1999). *IETF RFC 2660: The Secure Hypertext Transfer Protocol*. Obtido em 23 de Outubro de 2008, de The Internet Engineering Task Force: http://www.ietf.org/rfc/rfc2660.txt

Rishe, N., Sun, Y., Chekmasov, M., Andriy, S., & Graham, S. (2004). System architecture for 3D terrafly online GIS. In Proceedings of the IEEE sixth international symposium on multimedia software engineering. 273–276.

Robinett, W., & Holloway, R. (1992). Implementation of Flying, Scaling and Grabbing in Virtual Worlds. *Proceedings of the 1992 Symposium on Interactive 3D Graphics* , 189-192.

Schadow, G., & McDonald, C. (2008). Obtido em 2 de Maio de 2009, de The Unified Code for Units of Measure, version 1.7: http://aurora.regenstrief.org/~ucum/ucum.html

Schut, P., & Whiteside, A. (Setembro de 2005). OpenGIS Implementation Specification #05-007: OpenGIS Web Processing Service.

Shneiderman, B. (1987). *Designing the User Interface. Strategies for Effective Human-Computer Interaction.* Addison-Wesley.

Shumilov, S., Thomsen, A., Cremers, A. B., & Koos, B. (2002). Managment and visualization of large, complex and time-dependent 3D objects in distributed gis. *Proc. of the 10th ACM international symposium on Advances in geographic information systems* (pp. 113–118). ACM.

Siegel, A., & White, S. *The development of spatial representations of large-scale environments. In H. Reese (Ed.), Advances in Child Development and Behavior.* New York: Academic Press.

Sliuzas, R. (2003). Governance and the use of gis in developing countries. *Habitat International, 27 (4)* , pp. 495–499.

Stearns, J., Chinnici, R., & Sahoo. (Maio de 2006). *Update: An introduction to the java ee 5 platform*. Obtido em 26 de Abril de 2009, de Java ee at a glance: http://java.sun.com/developer/technicalArticles/J2EE/intro_ee5/

Steed, A. (2004). Data visualization within urban models. *Proc. of the Theory and Practice of Computer Graphics* (pp. 9–16). IEEE.

SUN Microsystems. (2008). *Java web start technology*. Obtido em 10 de Abril de 2009, de http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp

Sunday, D. (2004). *Polyline Simplification*. Obtido em 10 de Novembro de 2008, de http://softsurfer.com/Archive/algorithm_0205/algorithm_0205.htm

Tan, D., Robertson, G., & Czerwinski, M. (2001). Exploring 3D navigation: combining speed-coupled flying with orbiting. *ACM CHI'2001 Conference on Human Factors in Computing Systems* , 418-425.

The Apache Software Foundation. (2009). Obtido em 27 de Novembro de 2008, de Apache Tomcat: http://tomcat.apache.org

The Internet Corporation for Assigned Names and Numbers. (6 de Março de 2007). *MIME Media Types*. Obtido em 10 de Outubro de 2008, de Internet Assigned Numbers Authority: http://www.isi.edu/in-notes/iana/assignments/media-types/

The JUMP Unified Mapping Platform. (2003). Obtido em 23 de Janeiro de 2009, de http://www.jump-project.org

Thompson, H. S., Beech, D., Maloney, M., & Mendelsohn, N. (28 de Outubro de 2004). *XML Schema Part 1: Structures*. Obtido em 7 de Dezembro de 2008, de W3C Recommendation: http://www.w3c.org/TR/xmlschema-1

Thorndyke, P., & Hayes-Roth, B. (1982). Differences in Spatial Knowledge Acquired from Maps and Navigation. *Cognitive Psychology* , 560-589.

Tufte, E. (1990). *Envisioning Information.* Chesire, CT: Graphics Press.

Turner, A. (2006). *Introduction to Neogeography.* O'Reilly Media.

Uchoa, H. N., & Ferreira, P. R. (17 de Outubro de 2004). *Geoprocessamento com Software Livre*. Obtido em 14 de Janeiro de 2009, de http://www.geolivre.org.br/modules/mydownloads

uDig. (2008). *uDig - User-friendly Desktop Internet GIS*. Obtido em 22 de Fevereiro de 2009, de http://udig.refractions.net/

Universidade Federal do Estado do Rio de Janeiro. (2007). *GML - Geography Markup Language.* Obtido em 20 de Abril de 2009, de That's All Folks !: http://aiwww.wordpress.com/2007/05/04/gml-geography-markup-language/

University of Bonn. (2008). *Heidelberg 3D*. Obtido em 27 de Agosto de 2008, de 3D GDI Heidelberg: http://www.gdi-3d.de

Vinson, N. (1999). Design Guidelines for Landmarks to Support Navigation in Virtual Environments. *Proceeding of CHI '99* , 278-285.

Vretanos, P. (Maio de 2005). OpenGIS Implementation Specification #04-094: Web Feature Service Implementation Specification.

Vretanos, P. (2005). OpenGIS Implementation Specification #04-095: Filter Encoding Implementation Specification.

Ware, C. (2000). *Information Visualization: Perception for Design.* San Francisco, CA: Morgan Kaufmann Publishers.

Ware, C., & Franck, G. (1996). *Evaluating stereo and motion cues for visualizing information nets in three dimensions.* ACM Transactions on Graphics.

Ware, C., Plumlee, M., Arsenault, R., Mayer, L. A., & Smith, S. (2001). GeoZui3d: Data fusion for interpreting oceanographic data. *In Proceedings of oceans 2001* , 1960–1964.

Whiteside, A. (Fevereiro de 2007). OpenGIS Implementation Specification #06-121: OGC Web Services Common Specification.

Wikipédia. (2008). *Hiperligação*. Obtido em 11 de Setembro de 2008, de http://pt.wikipedia.org/wiki/Link

Wikipedia. (16 de Julho de 2008). *Styled Layer Descriptor*. Obtido em 2008 de Agosto de 12, de Wikipedia: http://en.wikipedia.org/wiki/Styled_Layer_Descriptor

Witmer, B., Bailey, J., Knerr, B., & Parsons, K. (2002). Virtual spaces and real world places: Transfer of route knowledge. *International Journal of Human-Computer Studies* , 413-428.

WKB4J. (2003). Obtido em 23 de Janeiro de 2009, de http://wkb4j.sourceforge.net

Wright, W. (1997). Business visualization applications. *Computer Graphics and Applications, IEEE , XVII*.

Wright, W., & Thomas, K. (2002). *Visualization of Blue Forces Using Blobology*. Monterey, California CA: Command and Control Research and Technology Symposium.

Zhou, M., Chen, M., & Feng, F. (2002). *Building a Visual Database for Example-Based Graphics Generation*. Washington, D.C.: In Proceedings of the 2002 IEEE Symposium on Information Visualization, IEEE Computer Society.

Zipf, A. (2008). Small island perspectives on global challenges: The role of spatial data in supporting a sustainable future. *GSDI 10 Conference Proceedings*.

Zlatanova, S., Rahman, A., & Pilouk, M. (2002). Trends in 3D GIS development. *Journal of Geospatial Engineering* , 1-10.