Universitat Autònoma
de Barcelona

Models de dades dels SIG a Internet. Aspectes teòrics i aplicats

Internet GIS Data models. Theoretical and applied aspects

Tesi Doctoral

Doctorat en Geografia

Facultat de Filosofia i Lletres

Universitat Autònoma de Barcelona

Doctorand: Joan Masó Pau

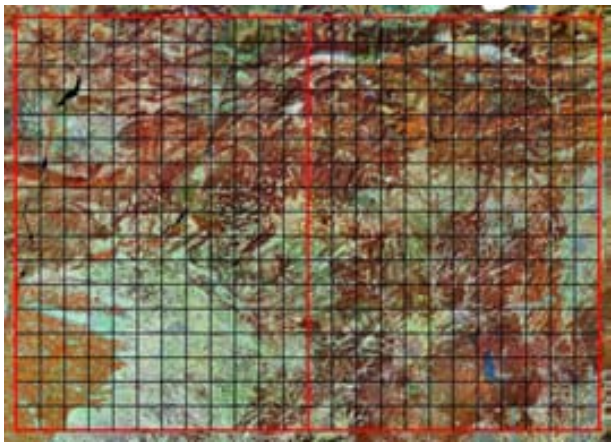
Director: Dr. Xavier Pons Fernández

Juny del 2012

La portada l'aquest document es basa en una composició RGB en color fals obtinguda amb 3 canals espectrals (Infraroig proper, infraroig mitjà i vermell) que provenen d'un mosaic d'escenes Landsat 7 preses l'agost de 2003 i que es pot obtenir del servidor SatCat.

(www.opengis.uab.cat/wms/satcat).

Per a realitzar la portada s'ha escollit una regió de Catalunya que no contingués mar ni zones fora de l'àmbit. Per a la contraportada (a l'esquerra) i per a la portada (a la dreta), la regió s'ha dividit en sengles parts



La imatge té 20 m de costat de píxel i ha estat tallada en tessel·les de 256x256 píxels amb els procediments de preparació de capes del MiraMon per accelerar el rendiment en proporcionar una capa en un servei conforme a l'estàndard WMTS. La tessel·lació consta de 55x55 tessel·les de les qual finalment es fa servir una matriu de 24x17, començant a la tessel·la (12,12) i acabant a la tessel·la (28,35).

“You're being confused by irrelevant data. Ignore it.”

Seven of Nine

from: Survival Instinct (1999) Star Trek Voyager series

“This was just a first step. In time you’ll take another. Small moves, Ellie. Small moves.”

Ted Arroway

from: Contact (1997) movie

ÍNDEX/TABLE OF CONTENTS

Índex general/Table of contents

ÍNDEX/TABLE OF CONTENTS vii

AGRAÏMENTS / ACKNOWLEDGEMENTS xiii

RESUM (català) xvii

SUMMARY (English) xxi

1. INTRODUCCIÓ/INTRODUCTION 1

1.A. Introducció (versió en català) 3

1.A.1. Introducció general 3

1.A.1.1. La necessitat de la cartografia digital i la problemàtica de la seva difusió. 4

1.A.1.2. El sistema client-servidor 5

1.A.1.3. La interoperabilitat i estàndards per a la distribució de dades espacials 6

1.A.1.4. Les infraestructures de dades espacials 9

1.A.1.5. L'Open Geospatial Consortium (OGC) 11

1.A.1.6. Estil d'arquitectura orientat a serveis 12

1.A.1.7. Estil d'arquitectura orientada a recurs 13

1.A.1.8. El SIG MiraMon 13

1.A.2. Motivació de la tesi 14

1.A.2.1. Problemàtiques conceptuals i d'arquitectura derivades de la implementació d'estàndards geospacials en infraestructures de dades distribuïdes. 15

1.A.2.1.1. Situació actual de les infraestructures de dades espacials 15

1.A.2.1.2. La opinió i contribució de l'usuari final (user feedback). 16

1.A.2.1.3. L'hipermapa i les seves quatre limitacions 17

1.A.2.2. Problemàtiques específiques de la navegació de mapes a Internet i casos d'aplicació 18

1.A.2.2.1. Anàlisi rigorosa del rendiment dels servidors de mapes 19

1.A.2.2.2. Mapes tallats en tessel·les 20

1.A.2.2.3. El format JPEG2000 21

1.A.3. Objectius de la tesi 23

1.A.4. Organització de la tesi 24

1.B. INTRODUCTION (English version) 27

1.B.1. General introduction 27
1.B.1.1. The need for digital cartography and its dissemination problems ... 27
1.B.1.2. The client-server system 29
1.B.1.3. Interoperability and the standards for distributing spatial data 30
1.B.1.4. Spatial data infrastructures 33
1.B.1.5. The Open Geospatial Consortium (OGC) 34
1.B.1.6. Service oriented architectural style 36
1.B.1.7. Resource oriented architectural style 36
1.B.1.8. The MiraMon GIS 36
1.B.2. Motivation of the thesis 38
1.B.2.1. Conceptual and architectural issues in implementing of standards for distributed spatial data infrastructures 38
1.B.2.1.1. Current status of the data infrastructures 38
1.B.2.1.2. End user feedback 39
1.B.2.1.3. The hypermap and its four limitations 40
1.B.2.2. Specific issues of Internet map browsers and uses cases 41
1.B.2.2.1. Rigorous analysis of map server performance 42
1.B.2.2.2. Maps cut into tiles 43
1.B.2.2.3. The JPEG2000 format 44
1.B.3. Thesis objectives 46
1.B.4. Document organization 47

2. TUNING THE SECOND GENERATION SDI: THEORETICAL ASPECTS AND REAL USE CASES 49
2.A.1. Introduction 51
2.A.1.1. SDI generations 53
2.A.1.2. User-side versus service-side improvements in the current SDI generation 53
2.A.1.3. Objectives and structure of the article 54
2.A.2. Review of SDI geoportal components 54
2.A.3. Improving the user and service sides in current SDI implementations ... 57
2.A.3.1. Improving metadata about data 58
2.A.3.2. Improving metadata about services 65
2.A.3.3. Improving data models 67
2.A.3.4. Improving data download 68

2.A.3.5. Improving data services and adding processing services 69
2.A.3.6. Improving data portrayal and symbolization 70
2.A.3.7. Adding mass market, VGI and Web 2.0 71
2.A.4. Use case 1: accessibility to healthcare centres 72
2.A.5. Use case 2: Web 2.0 user metadata comments: IDECTalk 75
2.A.6. Conclusions 78
2.A.7. Acknowledgements 78
2.A.8. References 78

3. BUILDING THE WORLD WIDE HYPERMAP (WWH) WITH A RESTFUL ARCHITECTURE 83
3.A.1. Introduction 85
3.A.2. Technological methodologies 87
3.A.2.1. Geospatial Web services and dynamically generated hyperlinks 87
3.A.2.2. Global geo-identifiers 88
3.A.2.3. Hyperlinks with purpose 89
3.A.2.4. RESTful Web services 90
3.A.3. How to implement the WWH 91
3.A.3.1. Catalogues in the WWH 97
3.A.3.2. Services in the WWH 97
3.A.4. Implementation example 98
3.A.4.1. Modifications in the GIS internal component 99
3.A.4.2. Corporative RESTful server 100
3.A.5. Conclusions 100
3.A.6. Acknowledgements 101
3.A.7. References 101

4. OPENGIS WEB MAP TILE SERVICE IMPLEMENTATION STANDARD 105
Contents 108
Foreword 117
Introduction 118
4.A.1. Scope 121
4.A.2. Compliance 121
4.A.3. Normative references 121
4.A.4. Terms and definitions 122

4.A.5. Conventions 123
4.A.5.1. Abbreviated terms 123
4.A.5.2. UML notation 124
4.A.5.3. Used parts of other documents 124
4.A.5.4. Platform-neutral and platform-specific standards 124
4.A.5.5. UML graphical and table representations 124
4.A.6. WMTS overview 126
4.A.6.1. Tile matrix set – the geometry of the tiled space 127
4.A.6.2. Well-known scale sets 130
4.A.7. WMTS Implementation model 131
4.A.7.1. Service metadata 131
4.A.7.1.1. ServiceMetadata document 132
4.A.7.1.2. GetCapabilities operation (mandatory in procedure oriented architectural style) 156
4.A.7.1.3. ServiceMetadata resource request (mandatory in resource oriented architectural style) 159
4.A.7.2. Tile 159
4.A.7.2.1. Tile resource 160
4.A.7.2.2. GetTile operation (mandatory in procedure oriented architectural style) 160
4.A.7.2.3. Tile resource request (mandatory in resource oriented architectural style) 164
4.A.7.3. FeatureInfo 164
4.A.7.3.1. FeatureInfo document 164
4.A.7.3.2. GetFeatureInfo operation (optional in procedure oriented architectural style) 165
4.A.7.3.3. FeatureInfo resource request (optional in resource oriented architectural style) 169
4.A.7.4. Operation request encoding 169
4.A.8. WMTS using HTTP KVP encoding 169
4.A.8.1. GetCapabilities 170
4.A.8.1.1. GetCapabilities request HTTP KVP encoding 170
4.A.8.1.2. GetCapabilities request HTTP KVP encoding example 170
4.A.8.1.3. GetCapabilities HTTP KVP encoding response 170
4.A.8.1.4. GetCapabilities HTTP KVP encoding response example 170

4.A.8.2. GetTile 171
4.A.8.2.1. GetTile request HTTP KVP encoding 171
4.A.8.2.2. GetTile request HTTP KVP encoding example 171
4.A.8.2.3. GetTile HTTP KVP encoding response 172
4.A.8.2.4. GetTile HTTP KVP encoding response example 172
4.A.8.3. GetFeatureInfo 172
4.A.8.3.1. GetFeatureInfo request HTTP KVP encoding 172
4.A.8.3.2. GetFeatureInfo request HTTP KVP encoding example 173
4.A.8.3.3. 8.3.3 GetFeatureInfo HTTP KVP encoding response 173
4.A.8.3.4. GetFeatureInfo HTTP KVP encoding response example 173
4.A.8.4. Exceptions in HTTP KVP encoded operations 174
4.A.9. WMTS using SOAP encoding 174
4.A.9.1. GetCapabilities 174
4.A.9.1.1. GetCapabilities request SOAP encoding 174
4.A.9.1.2. GetCapabilities request SOAP encoding example 174
4.A.9.1.3. GetCapabilities SOAP encoding response 174
4.A.9.1.4. GetCapabilities SOAP encoding response example 175
4.A.9.2. GetTile 175
4.A.9.2.1. GetTile request SOAP encoding 175
4.A.9.2.2. GetTile request SOAP encoding example 175
4.A.9.2.3. GetTile SOAP encoding response 176
4.A.9.2.4. GetTile SOAP encoding response example 176
4.A.9.3. GetFeatureInfo 177
4.A.9.3.1. GetFeatureInfo request SOAP encoding 177
4.A.9.3.2. GetFeatureInfo request SOAP encoding example 177
4.A.9.3.3. GetFeatureInfo SOAP encoding response 177
4.A.9.3.4. GetFeatureInfo SOAP encoding response example 178
4.A.9.4. Exceptions in SOAP encoding 179
4.A.10. WMTS using RESTful 180
4.A.10.1. ServiceMetadata resource (mandatory in resource oriented architectural style) 181
4.A.10.1.1. GetResourceRepresentation request 181
4.A.10.1.2. GetResourceRepresentation request example 181
4.A.10.1.3. ServiceMetadata representation 181

4.A.10.1.4. ServiceMetadata representation example 181
4.A.10.1.5. GetResourceRepresentation exception 181
4.A.10.2. Tile resource (mandatory in resource oriented architectural style) 182
4.A.10.2.1. GetResourceRepresentation request 182
4.A.10.2.2. GetResourceRepresentation request example 184
4.A.10.2.3. Tile representation 184
4.A.10.2.4. Tile representation example 184
4.A.10.2.5. GetResourceRepresentation exception 185
4.A.10.3. FeatureInfo resource (optional in resource oriented architectural style) 185
4.A.10.3.1. GetResourceRepresentation request 185
4.A.10.3.2. GetResourceRepresentation request example 188
4.A.10.3.3. FeatureInfo representation 188
4.A.10.3.4. FeatureInfo representation as an XML document example ... 188
4.A.10.3.5. GetResourceRepresentation exception 188
4.A.11. Recommendations to improve interoperability and performance. ... 188
4.A.11.1. Server and Client support for KVP, SOAP and RESTful 188
4.A.11.2. A standard set of scales 189
4.A.11.3. A standard image format and FeatureInfo document response 189
4.A.11.4. Number of TileMatrixSets and TileMatrixSetLimits 189
4.A.11.5. 11.5 Cacheble resources 189
Annex A: Abstract test suite 191
Annex B: XML Schema Documents 206
Annex C: UML model 208
Annex D: Example XML documents 217
Annex E: Well-known scale sets 222
Annex F: WSDL description of the service 226
Annex H: Pseudocode 232
Bibliography 234

5. COMBINING JPEG2000 COMPRESSED FORMATS AND OGC STANDARDS FOR FAST AND EASY DISSEMINATION OF LARGE SATELLITE DATA 235
5.A.1. Introduction 238
5.A.2. Integration in OGC Standards 240
5.A.2.1. Using JPEG2000 in SOS 240

5.A.2.2. Using JPEG2000 and JPIP in WCS 240
5.A.2.3. Using JPEG2000 in WMS and WMTS services 242
5.A.2.4. Comparing WMTS and JPIP services 247
5.A.2.5. GMLJP2 248
5.A.3. Conclusions 248
5.A.4. Acknowledgements 249
5.A.5. References 249

6. IMPACT OF USER CONCURRENCY IN COMMONLY USED OPEN GEOSPATIAL CONSORTIUM MAP SERVER IMPLEMENTATIONS 251
6.A.1. Introduction 253
6.A.2. Materials and methodology 253
6.A.3. Evaluation of concurrent requests to a single server 254
6.A.4. Evaluation of a cluster of servers 255
6.A.5. Tiling the request and response 256
6.A.6. Conclusions 259
6.A.7. Acknowledgements 259
6.A.8. References 259

7. RESUM DE RESULTATS / SUMMARY OF RESULTS 261
7.A. Resum de resultats (versió en català) 263
7.A.1. Aspectes generals derivats de la implementació d'estàndards geospacials 263
7.A.1.1. Aspectes conceptuals i d'arquitectura 263
7.A.1.1.1. Resultats de l'anàlisi de les Infraestructures de dades espacials.... 263
7.A.1.1.2. Fent Evolucionar el concepte de l'hipermapa: L'hipermapa mundial. 268
7.A.1.2. Casos d'aplicació 272
7.A.1.2.1. La web 2.0 i els comentaris dels usuaris sobre les metadades: IDECTalk 272
7.A.1.2.2. MiraMon-REST 273
7.A.2. Aspectes específics de la navegació de mapes a Internet 275
7.A.2.1. Aspectes conceptuals i d'arquitectura 275
7.A.2.1.1. Servei web de mapes tessel·lats (WMTS) 275
7.A.2.1.2. Ús del format JPEG2000 en servidors de mapes 277
7.A.2.2. Casos d'aplicació 280

7.A.2.2.1. Anàlisi de rendiment dels servidors del mapes..... 280
7.B. Summary of results (English version) 283
7.B.1. Generic aspects derived from geospatial standards implementation ... 283
7.B.1.1. Conceptual and architectural aspects..... 283
7.B.1.1.1. Results from the spatial data Infrastructures analysis. 283
7.B.1.1.2. Evolving the hypermap concept: The World Wide Hypermap. 288
7.B.1.2. Use Cases..... 291
7.B.1.2.1. Web 2.0 and metadata user feedback: IDECTalk 291
7.B.1.2.2. MiraMon-REST..... 293
7.B.2. Specific aspects of the Internet map browsing 295
7.B.2.1. Conceptual and architectural aspects..... 295
7.B.2.1.1. Web Map Tile Service (WMTS) 295
7.B.2.1.2. Using JPEG2000 in map services 296
7.B.2.2. Use cases 299
7.B.2.2.1. Maps server performance analysis..... 299
8. CONCLUSIONS 303
8.A. Conclusions (versió en català) 305
8.B. Conclusions (English version)..... 309
BIBLIOGRAFIA/REFERENCES 313
ANNEX I: FUNDAMENTOS DE LAS IDE. OPEN GEOSPATIAL CONSORTIUM (OGC) 323
ANNEX II LIST OF RESOURCES AND OPERATIONS IN THE WWH 337
ANNEX III ACRÒNIMS / ACRONYMS 345

Figures (versió en català)

Figura 1: Sistema client - servidor i protocol de comunicacions (font: elaboració pròpia). 6
Figura 2: Interoperabilitat d'un programari client amb diversos servidors de mapes (font: elaboració pròpia). 9
Figura 3: Components bàsics de les IDE de segona generació (font: elaboració pròpia). 10
Figura 4: Exemples de portals VGI: (a): GeoWiki (font: http://www.geo-wiki.org/) (b): MetOfficeWOW (font: http://wow.metoffice.gov.uk/). 17

Figura 5: Estàndards de visualització (font: figura 2, capítol 2). 19

Figura 6: Els clients WMS utilitzen generalment les operacions GetMap per a demanar al servidor tota l'àrea de pantalla de la vista (font: elaboració pròpia)..... 19

Figura 7: Un client de tessel·les demana peticions simultànies de totes les que cobreixen l'àrea de la vista, excepte en el cas que les tingui en *caché* del client. També oculta les parts de tessel·la que no entren en la vista (font: elaboració pròpia)..... 20

Figura 8: (a) Definició del patró de tall emprat (font: figura 2, capítol 4) i (b) nivells de zoom per a un dels sistemes de mapes en tessel·les, en aquest cas l'estàndard OGC anomenat WMTS (font: figura 3, capítol 4)..... 21

Figura 9: Transformada *wavelet* d'una composició en fals color d'un fragment d'imatge Landsat de l'àrea metropolitana de Barcelona. En línia continua, 3 nivells de resolució de la transformada. Amb línies de punts: divisió en *codeblocks* (només els 2 primers *codeblocks* són necessaris per a recuperar una imatge completa de resolució ¼ de la resolució original) (font: elaboració pròpia amb el programa fwt2d). 22

Figura 10: La imatge original (a) es pot dividir en tessel·les (b) que són transformades *wavelet* (c), tallades en *codeblocks* i incorporades a la seqüència de bytes del fitxer JPEG2000 (d) (figura basada en: figura 7, capítol 5). 23

Figura 11: Estàndards potencialment utilitzats a les IDE classificats pel seu paper (font: figura 2, capítol 2)..... 264

Figura 12: Relacions entre els diferents elements de les metadades sobre les dades, les metadades sobre serveis, i els estàndards de models de dades; a vegades realitzades per enllaços URL, a vegades per identificadors (font: elaboració pròpia). 265

Figura 13: Entitat responsable de mantenir la unicitat de cadascun dels 3 nivells d'una URI de recurs dins del WWH (font: elaboració pròpia)..... 269

Figura 14: Resum dels recursos i les seves operacions, les relacions i les plantilles URI en el WWH (font: figura 2, capítol 3). 270

Figura 15: Arquitectura de l'IDECTalk. El *mash-up* 1 es connecta a la IDEC amb un UUID i el *mash-up* 2 es connecta amb el Google Maps utilitzant l'envolupant (font: figura basada en la figura 5 del capítol 2)..... 273

Figura 16: Components de programari i les connexions entre 2 nodes corporatius i un client lleuger extern en el WWH (font: figura 3, capítol 3). 274

Figura 17: Dos TileMatrixSets diferents formats per diverses TileMatrix que utilitzen conjunts d'escales diferents (3 són visibles a la figura). (a) està usant els WKSS *GoogleMapsCompatibleWKSS* (les tessel·les provenen de l'OpenStreetMap); on cada TileMatrix es divideix en quadrats de mosaics regulars. (b) no segueix una WKSS i utilitza tessel·les rectangulars (font: elaboració pròpia)..... 276

Figure 18: Tessel·les a 3 nivells d'escala i correspondència entre *codeblocks* a 3 nivells de resolució de la transformada *wavelet*: en color taronja es representen les equivalències en el nivell de resolució de base, en color vermell equivalències en el segon nivell de resolució, i en color groc les equivalències en el tercer nivell de resolució (incomplet per raons de llegibilitat) (font: elaboració pròpia)..... 277

Figura 19: Un sol repositori de JPEG2000 internament en tessel·les (amb les metadades codificades usant GMLJP2) pot server per SIG interns d'escriptori, per a servidors WCS i per a servidors WMTS (font: figura 9, capítol 5). 279

Figura 20: Temps de resposta de les distintes sol·licituds simultànies amb un màxim de 17 clients a un servidor WMS del MiraMon: (a) configuració de servidor únic (font: figura 4, capítol 6), (b) configuració en *cluster* de 6 servidors (font: figura 5, capítol 6). 281

Figures (English version)

Figure 1: Client-server system and the communication protocol (source: own preparation)..... 30

Figure 2: Interoperability of a client software with multiple map servers (source: own preparation)..... 32

Figure 3: Basic components of the second generation of SDI (source: own preparation). 34

Figure 4: VGI portal examples: (a): GeoWiki (font: http://www.geo-wiki.org/) (b): MetOfficeWOW (font: http://wow.metoffice.gov.uk/). 40

Figure 5: Visualization standards (source: Figure 2, chapter 2). 41

Figure 6: WMS clients commonly use GetMap operations to request the whole view-port from the server (source: own preparation)..... 42

Figure 7: A tile client makes simultaneous requests for all tiles that cover the view-port, except when they are already in the client *cache*. It also hides the tile areas that are not part of the view-port (source: own preparation). 43

Figura 8: (a) Tile pattern definition (source: Figure 2, chapter 4) and (b) zoom levels for a map tile system, in this case the WMTS OGC standard (source: Figure 3, chapter 4).44

Figure 9: *Wavelet* transform of a false colour composition of a fragment of a Landsat image of the Barcelona metropolitan area. Solid line, three transform resolution levels. Dashed line: *codeblocks* division (only the first two *codeblocks* are required to decompress the whole image to ¼ of the original de resolution) (source: own preparation using fwt2d software). 45

Figure 10: The original image (a) can be cut into tiles (b) that are independently *wavelet* transformed (c), cut into *codeblocks* and incorporated in the byte sequence in the JPEG2000 file (d) (Figure based on: Figure 7, in chapter 5). 46

Figure 11: Standards potentially used in SDI classified by its role (source: Figure 2, chapter 2). 284

Figure 12: Relationships between different elements in data metadata, service metadata, and model standards, sometimes made by URL links, sometimes by identifiers (source: own preparation). 285

Figure 13: Entity responsible for maintaining the uniqueness of each of the 3 parts of a resource URI in the WWH (source: own preparation). 289

Figure 14: Summary of resources and their operations, relations and URI templates in the WWH (source: Figure 2, chapter 3). 290

Figure 15: IDECTalk architecture. Mash-up 1 connects to the Catalan SDI using a UUID and mash-up 2 connects to Google Maps using the Bounding Box (Figure based on Figure 5 in chapter 2). 292

Figure 16: Software components and connections between two corporate nodes and an external thin client in the WWH (source: Figure 3, chapter 3). 293

Figure 17: Two different TileMatrixSets composed by TileMatrix's that uses different scales sets (3 visible in the figure). (a) is using the GoogleMapsCompatibleWKSS (tiles come from OpenStreetMap); each TileMatrix is divided in squared regular tiles. (b) is not following a WKSS and uses rectangular tiles (source: own preparation). 296

Figure 18: Tiles at 3 scale levels and correspondence between *codeblocks* in a 3 resolution levels of the wavelet transform: orange represents equivalences in the base resolution level, red equivalences in the second resolution level and yellow equivalences in the third resolution level (incomplete for readability reasons, source: own preparation). 297

Figure 19: Single tiled JPEG2000 images repository (with metadata encoded in GMLJP2) for either internal GIS desktop, WCS server and WMTS server. (source: Figure 9, chapter 5) 298

Figure 20: Time response for different concurrent requests for up to 17 clients to a MiraMon WMS server: (a) single server configuration (source: Figure 4, chapter 6), (b) cluster of 6 servers configuration (source: Figure 5, chapter 6). 300

AGRAÏMENTS / ACKNOWLEDGEMENTS

Fa molt, molt de temps, en un departament no massa llunyà vaig començar un periple que hauria de durar 20 anys. Han passat massa anys però si no hagués estat així no podria explicar-vos aquest atípics agraïments en forma d'història.

En acabar la carrera la Núria Barniol i el Francesc Pérez em varen ensenyar què representava fer recerca i com es deixava de ser alumne per a passar a ser professor (ajudant, associat...) del grup d'electrònica del Departament de física. A ells els dec haver publicat el meu primer article científic i haver dut a terme la tesina, ambdues coses sobre microscòpia d'efecte túnel (res a veure amb el que parlarem en les properes planes).

Per aquelles dates vaig fer l'objecció de consciència a la UAB (per intentar no trencar la meva vida per la meitat amb allò que es deia la "mili") i vaig acabar a la Unitat de botànica on en Josep Maria Roure em va presentar el Xavier Pons. En aquell moment, ni en Xavier Pons ni jo no sabíem que tot allò seria un punt d'inflexió que conduiria, entre moltes altres coses, a aquesta tesi. De moment, vaig jugar a passar els dies d'objecció programant algunes eines vectorials topològiques per una cosa a la qual en Xavier Pons en deia SIG.

Un any després d'acabar la tesina, vaig haver d'abandonar el microscopi per intentar demanar un projecte Petri. Així, en Jordi Bartrolí em va ensenyar com es demanava un projecte científic de transferència junt amb una empresa del Vallès que feia recerca en espectroscòpia de la resistència de substàncies químiques. També em va ensenyar com es feia servir una biblioteca de revistes científiques; era l'època en què les revistes es publicaven només en paper i s'enquadernaven en grans volums estibats acuradament en llargues prestatgeries. No ens varen concedir el projecte.

Llavors va ser quan en Xavier Pons va pensar que allò del SIG tindria un nom: es diria MiraMon i el podia continuar ajudant per anar-lo fent créixer i em va presentar al Jaume Terradas i vaig acabar al CREAF. L'ambient familiar que s'hi respirava es una de les raons per les que mi he quedat de gust. L'Arnald Marcer, el Jordi Valeriano, el Lluís Pesquer i el Carles Dalmases es varen anar incorporant progressivament a un projecte que anava creixent. La nostàlgia de temps passats em va fer fer un segon mestratge en enginyeria electrònica amb en Xavier Aymerich com a tutor.

En Xavier Pons sempre va creure en els meus invents. Va entendre la necessitat del web del CREAF i la vàrem aprofitar per muntar una enquesta *on-line* força abans que ningú no parlés de l'ADSL (quantes hores de mòdem!). Ell va proposar la idea de l'MMZ que es va fer en un estiu calorós. També va tolerar la meva idea del web del MiraMon amb compra i tot (com si nosaltres fóssim el Corte Inglés!). Quasi per casualitat vàrem ser acceptats per un projecte i2cat tot proposant de fer un navegador de mapes, i a mig projecte varem descobrir el WMS i l'OGC i ho vàrem reorientar. M'oblidava de tantes hores invertides en la lectura dels estàndards de metadades amb l'Antònia Valentín i Alaitz Zabala. Després va venir l'Edu Luque i la Núria Julià, l'Abel Pau, la Ivette Serral (la seva inconsciència fou fonamental per demanar el GeoViQua), l'Ester Prat (i els estudis d'accessibilitat a centres de salut) i el Xavier Calaf. Amb la Núria Julià

hem discutit continguts dels estàndards (vegeu capítol 4) i hem passat moltes hores consolidant els navegadors i servidors de mapes.

Hem de fer un parèntesis a la història per destacar un dels factors externs que més han influït en aquest treball: el grup de gent que participa a l'OGC. Amb ells he aprés a parlar millor l'anglès, a fer teleconferències, a discutir sobre estàndards, a participar en desenvolupaments experimentals compartits, però també he vist com es dirigeix un grup de treball i com s'estimula la creació. Me'ls trobo per tots els congressos internacionals, a les discussions de la ISO...

La historia continua amb la gent de Geografia, la càtedra d'en Xavier Pons i els Grumets i la transformació del CREAF iniciada pel Ferran Rodà i accelerada pel Javier Retana, i la necessitat de consolidar la recerca que implícitament s'havia fet amb tants anys de picar codi C. Ara es quan començaria la redacció de les publicacions recollides en aquest compendi. Tornar a provar d'obrir-se camí en la recerca no ha estat fàcil i vull agrair especialment al Xavier Pons les converses d'un contingut que em fa vergonya confessar aquí, però que podem resumir en que ell ha confiat més en les meves capacitats que jo mateix. També ha ajudat molt l'Alaitz Zabala tant en la seva participació en els articles (vegeu capítol 2, 3 i 5) com en que hagi obert i mostrat el camí amb la seva relativament recent presentació de tesi. I ara també hi ha la Paula Díaz que té molta empenta, constància i potencial (vegeu capítol 6). I això només és el començament d'un camí que no sé on porta (*Small moves, Ellie. Small moves*) i que voldria poder continuar en el nostre grup.

Obviously, this document and the research made by the whole group would not been possible without the economical support of the Geography department, the CREAF, the Spanish Government and FEDER funds (under grants TSI2006-14005-C02-02, TIN2009-14426-C02-02), the Catalan Government (grant to Consolidated Research Groups [2009 SGR 1511]), the European Commission (under grants FP7-242390-GEO-PICTURES project [FP7-SPACE-2009-1], FP7-265178-GeoViQua [ENV.2010.4.1.2-2] and FP7-265124-EGIDA [ENV.2010.4.1.2-1]) and the Institut Cartogràfic de Catalunya (that funded the Interoperability Framework Report for the Catalan Cartographic Plan).

I durant tot aquest temps, sempre amb la Marta i després l'Elisabet i en Joan, i ara en Jordi, que només entén que el seu pare es passa moltes hores amb el "dnador". I els meus pares i familiars que m'han perdut de vista i encara es pregunten si l'estil de vida que he portat, prou diferent del que ells varen recórrer, té algun sentit. Jutgeu-ho vosaltres.

RESUM (català)

Existeix abundant literatura científica sobre els estàndards d'informació geogràfica, i particularment relacionats amb les infraestructures de dades espacials. Malgrat això, pocs treballs descriuen les implicacions de la seva implementació, així com les dificultats d'aquesta i del seu desplegament en l'àmbit de la cartografia i de la teledetecció. Aquesta tesi analitza l'abast d'aplicació dels estàndards internacionals, detecta els problemes d'encavalcament i de falta de definició en alguns aspectes, i realitza propostes per tal de corregir aquests problemes tant en les implementacions com en els propis estàndards. Un dels problemes observats és la desconnexió entre els diferents estàndards i, per aquest motiu, es proposa la definició d'un nou marc tecnològic que permet reestructurar l'estil d'arquitectura emprada en les infraestructures de dades espacials, basat en una orientació a servei, tot canviant-lo per una altre estil d'arquitectura orientat a recurs amb la finalitat de fer un millor ús de la pròpia Internet, de relacionar millor els recursos entre ells fins a crear un gran hipermapa mundial que anomenen World Wide Hypermap (WWH), que incorpora altres col·lectius com són, per exemple, aquells que s'adhereixen a polítiques de dades obertes (*open data*), la cartografia generada per voluntariat i els navegadors de mapes pel mercat de masses i els globus virtuals. Durant l'elaboració d'aquests treballs s'ha participat activament en el procés de definició d'estàndards internacionals i s'ha contribuït liderant l'edició del Web Map Tile Service (WMTS) que va ser ratificat com a estàndard per l'Open Geospatial Consortium (OGC). Al mateix temps, s'ha posat a prova tecnologies amb dades reals, cosa que ha permès detectar tant problemes en les especificacions com en les seves implementacions, el que fa que s'hagi pogut proposar correccions a aquestes especificacions quan ha semblat convenient. Així mateix, en la tesi s'analitza quantitativament els diferents productes de serveis de mapes de diferents fabricants (incloent les implementacions realitzades en el MiraMon) i es valora fins a quin punt les millores proposades contribueixen a millorar el rendiment de les implementacions. En aquest sentit, la tesi fa propostes concretes per emprar el JPEG2000 en implementacions de serveis que integrin simultàniament Web Map Service (WMS), Web Map Tile Service (WMTS) i Web Coverage Service (WCS). També s'ha incorporat les especificacions i solucions al programari MiraMon (tant en la versió d'escriptori com en els servidors i navegador de mapes) per tal de validar-ne el seu ús, i s'ha dissenyat sistemes que permetin l'optimització dels algorismes que implementen les anteriors propostes, alhora que s'ha fet propostes d'implementació de l'aproximació REST al MiraMon. Finalment s'ha realitzat implementacions concretes per a incloure les contribucions dels usuaris finals com a informacions complementàries a les tradicionalment presents als catàlegs de metadades.

SUMMARY (English)

There is abundant scientific literature on geographical information standards, particularly in relation to spatial data infrastructures. However, there are only a few studies on the implications of implementing them and the difficulties involved, or on their use in cartography and the remote sensing fields. This thesis analyses the scope of applying international standards, and identifies some overlaps and gaps in certain areas. Proposals are also made to fix the problems in both the implementations and the standard documents themselves. One problem we observed is that the different standards are not completely connected. Therefore, we propose defining a new technological framework that makes it possible to restructure the architectural style used in the spatial data infrastructures, which is currently service oriented. We propose changing it to a resource oriented architectural style, which would improve the use of the Internet and relate resources better in a way that a single World Wide Hypermap (WWH) is created. The WWH incorporates other collectives better, such as those that join the open data policies and standards, the volunteered geographic information groups, the mass market map browsers and the virtual globes. While carrying out this work we have been actively involved in the process of defining international standards, which has led to the development of the Web Tile Map Service (WMTS) document that has been ratified as a standard by the Open Geospatial Consortium (OGC). At the same time, we have tested technological implementations with real data in order to detect problems in the specification documents and in their implementations. Adjustments to these specifications have been proposed when appropriate. Furthermore, this PhD thesis quantitatively analyses different products from different map service providers (including MiraMon implementations) and assesses the extent to which the proposed improvements contribute to increasing the performance of the implementations. In this regard, this document makes a specific proposal for using JPEG2000 in Web service implementations that simultaneously integrates the Web Map Service (WMS), Web Map Tile Service (WMTS) and Web Coverage Service (WCS). We have also incorporated some of the proposed specifications and solutions into the MiraMon software (desktop, map browser and map service) in order to test and validate them. Moreover, we have designed some ways of optimizing the algorithms that implement previous proposals, and an implementation of the REST approach has been suggested for MiraMon. Finally, we propose a specific implementation that includes end-user contributions (user feedback), so that they complement the current metadata catalogues.

1. INTRODUCCIÓ/INTRODUCTION

1.A. Introducció (versió en català)

1.A.1. Introducció general

La introducció d'Internet, primer a les universitats i posteriorment a les empreses i altres nivells educatius fins arribar a l'àmbit domèstic, ha propiciat que tots els ordinadors, que fins el moment havien estat disgregats, s'hagin pogut interconnectar entre ells per tal de generar un flux de dades i serveis que potencia l'intercanvi immediat d'informació i experiències dels usuaris en un procés que s'ha anomenat "connexió a xarxa" i que ha estat bàsicament vehiculat pel correu electrònic i la web. Des dels inicis d'aquest procés els usuaris i productors de cartografia digital varen veure l'enorme potencial d'intercanvi que els suposava la xarxa per tal de treballar plegats o per tal de difondre o compartir la seva informació i reduir redundàncies (Meeksa *et al.*, 2004). Ràpidament el nombre d'usuaris de la cartografia digital es va disparar ja que els usuaris finals podien arribar a ella de manera fàcil i sense intermediaris; un procés que encara es va accelerar més amb la irrupció de les eines produïdes pels grans motor de cerca d'Internet (p.ex., MapQuest, la guia Michelin, i posteriorment Google Maps, Bing Maps, etc) i els globus virtuals (Butler, 2006; Sheppard *et al.*, 2004). Aquest procés no ha estat exempt de problemes, com ara la falta d'estandardització en els formats, la de programari de visualització amb simbolització intel·ligible, el drets d'autor, l'elevat cost de la cartografia (taxes d'ús) en alguns àmbits i països, la lentitud dels intercanvis de dades, la falta de formació dels usuaris, etc (Tu *et al.*, 2004; Yang *et al.*, 2005). Recentment aquest procés de canvi de paradigma s'ha accelerat encara més amb l'aparició dels núvols ordinadors on la capacitat d'emmagatzematge i de procés també s'està desplaçant de l'ordinador local a la xarxa però aquest procés és tan recent que resulta difícil de predir on ens portarà (Yang *et al.*, 2011).

Els organismes d'estandardització (World Wide Web Consortium [W3C], International Organization for Standardization / Technical Committee 211 [ISO/TC211], Open Geospatial Consortium [OGC], etc) estan treballant activament en l'establiment d'especificacions que permetin millorar l'experiència dels usuaris tot garantint la interoperabilitat dels procediments i formats emprats, al mateix temps que els proporcionin les funcionalitats necessàries (Albrecht, 1999; Percivall, 2010). D'altra banda, els organismes cartogràfics s'han organitzat a l'entorn d'Infraestructures[1] de Dades Espacials (IDE), que són organismes que tenen com a finalitat fomentar el descobriment, l'avaluació, l'accés, la distribució i la utilització de dades geogràfiques a diferents nivells. Per tal de poder acomplir aquests objectius, defensen, estimulen i

[1] La paraula *infraestructura* ha canviat de grafia des dels anys 90. Anteriorment s'escrivia com *infrastructura* (Diccionari de la Llengua Catalana de l'Enciclopèdia Catalana, edició 1998). Encara que no compartim les raons d'aquest canvi, el respecten i l'incorporen en aquest document tal com ha fet la IDEC. Aquest canvi va suscitar polèmiques entre els lingüistes. Així, el document inicial, de títol *Sobre la grafia dels compostos i derivats de mots que presenten etimològicament una essa inicial seguida de consonant*, va ser aprovat per la Secció Filològica el 17 de gener de 1992, però va haver de ser revisat i aprovat en un nou document de la Secció Filològica el 19 de gener de 1996, en aquest cas de títol *Sobre la grafia dels mots compostos i prefixats que contenen formants amb una essa inicial etimològica seguida de consonant*.

coordinen l'adopció dels estàndards[2]. Malgrat les positives finalitats d'aquests organismes i la seva considerable influència en el sector, el procés d'estandardització es troba lluny del seu acompliment (Crompvoets *et al.*, 2004). Per complicar la situació encara més, darrerament han aparegut alguns actors independents com el Google Maps/Earth o els sistemes GPS d'automoció, de nou allunyant dels estàndards en favor d'interessos empresarials d'exclusivitat.

1.A.1.1. La necessitat de la cartografia digital i la problemàtica de la seva difusió.

L'aparició de la cartografia digital va representar, i representa, molts avantatges respecte la cartografia tradicional en paper (Pons, 1996; Heywood *et al.*, 2006):

- La informació no espacial es guarda com a dades originals (generalment en taules) i no com una característica de simbolització de l'objecte espacial (un color, un gruix, etc).
- La manera com estan simbolitzades les entitats es pot canviar fàcilment per tal d'emfasitzar altres aspectes de les dades.
- Es pot transformar la informació (per exemple canviar la projecció o aplicar una generalització o un canvi d'escala).
- Es pot treballar amb entitats individualitzades que tinguin un identificador únic (vegeu capítol 3).
- Es pot establir vincles entre entitats i amb informació externa (vegeu capítol 3).
- Un conjunt d'informació geogràfica es pot mantenir actualitzat més fàcilment.
- Es poden superar les limitacions 2D del paper i treballar en models multidimensionals (per exemple, cartografia meteorològica 3D) o amb l'evolució temporal (per exemple, seguiment de flotes).
- No cal tallar en fulls els mapes de gran extensió i nivell de detall (tot i que per tradició o per limitacions en els formats, de vegades, també és necessari en la cartografia digital).
- La informació es pot transmetre, duplicar i imprimir fàcilment.
- Encara que en la cartografia tradicional es pot arribar a superposar informació i analitzar distàncies i algunes relacions entre entitats, la cartografia digital permet augmentar molt el ventall d'operacions analítiques i automatitzar anàlisis sobre grans volums de dades. Així, es poden provar hipòtesis alternatives amb més facilitat i amb criteris científics i repetibles.
- No es degrada amb el seu ús.

Per a fer servir la cartografia digital necessitem un sistema informàtic que ens faciliti la seva visualització, consulta i explotació. Aquests sistemes informàtics han anat variant al llarg del temps tot començant pels ordinadors de sobretaula que incorporen programari SIG, i continuant amb els dispositius mòbils amb sistemes de navegació i els sistemes de portals de mapes per Internet. Així, la cartografia digital es presenta en gran nombre de formats i sistemes, i precisament els primers problemes apareixen quan hi ha necessitat d'intercanvi d'informació. El problema es torna més evident quan

[2] Encara que hi ha una certa tendència a traduir la paraula anglesa *standard* per *norma*, en aquesta tesi preferim emprar el terme *estàndard* que està completament acceptat per l'enciclopèdia catalana.

els sistemes es troben interconnectats entre ells per xarxes de comunicacions i necessitem emprar dades de diferents orígens. I per a facilitar l'intercanvi d'informació apareixen els serveis de dades.

La cartografia digital agrupa la informació en conjunts d'informació (*datasets*) que utilitzen bàsicament dos models de dades: Un model de dades orientat a descriure variables que cobreixen de manera continua el territori i on els valors que d'aquestes es recullen també de forma quasi-contínua sobre aquest (*e.g.*, la temperatura, l'elevació, etc), i un altre model de dades orientat a identificar entitats sobre el territori, els quals es poden descriure de manera individual (per exemple, les unitats administratives, els edificis, les carreteres, etc). El primer model s'anomena de cobertes (*coverages*) i donarà lloc als serveis de distribució de cobertes (*coverage services*), mentre que el segon model s'anomena d'entitats (*features*) i donarà lloc als serveis de distribució d'entitats (*feature services*)[3]. Aquests models, tenen com a propòsit l'emmagatzematge estructurat i el tractament de la informació, però no són directament visualitzables perquè per a dur a terme aquesta tasca cal simplificar-los, associar-los un conjunt de regles de simbolització que proporcionin color, textura, petits textos, etc als valors o a les entitats, i adaptar-los a un dispositiu de visualització (com una pantalla o una impressora) per tal de generar un mapa; aquests darrers procediments, en un entorn distribuït, poden ser delegats a serveis de mapes (*map services*). Quan aquests serveis es donen emprant la web, s'anomenen Web Coverage Service (WCS) (Baumann, 2010), Web Feature Service (WFS) (Vretanos, 2010) i Web Map Service (WMS) (de la Beaujardiere, 2004) (o Web Map Tile Service[WMTS] [Masó *et al.*, 2010a]) respectivament.

Cada conjunt d'informació representa una tema concret (per exemple: hidrografia, edificacions, humitat ambiental, densitat de població, etc). El nombre de conjunts d'informació que un sistema informàtic pot emmagatzemar pot ser molt gran, el que obliga a ordenar aquests conjunts d'informació per ter la seva temàtica, escala, època, etc. Aquesta classificació es més o menys arbitraria i cada organisme utilitza la seva pròpia aproximació de forma que quan els diferents magatzems d'informació s'uneixen entre ells, aquesta classificació ja no resulta possible i apareix la necessitat de sistemes automàtics de cerca d'informació anomenats catàlegs de metadades que fan servir petites descripcions estructurades en forma de fitxa per a cada conjunt d'informació i donen lloc als serveis de catàleg (*catalogue services*). Aquests catàlegs permeten descobrir la informació present en el sistema a partir de petites interrogacions en un llenguatge més o menys natural. Quan aquests serveis es proporcionen emprant la web s'anomenen Catalogue Service Web (CSW) (Nebert, 2007).

Aquesta tesi estudia com millorar els diferents serveis i estàndards que s'apliquen per difondre cartografia digital.

[3] Per a simplificar, podem dir que el model orientat a cobertes fa servir habitualment una representació ràster (on les peces individuals són celles més o menys regularment distribuïdes en l'espai que habitualment contenen els valors d'una magnitud mesurada) i el model orientat a entitats fa servir habitualment una representació vectorial (on les peces individuals són generalment punts, línies i polígons lligats a llistes d'atributs no espacials), però no sempre es així.

1.A.1.2. El sistema client-servidor

La separació client-servidor és una de les bases de la computació distribuïda. En aquesta model, el sistema informàtic es separa en dos programes independents que generalment resideixen en ordinadors diferents (Peng *et al.*, 2003) (figura 1). El programa client fa una petició al programa servidor remot, el programa servidor fa la feina que el client ha demanat i envia el resultat de tornada cap al client. L'usuari del sistema generalment no interactua directament amb el servidor sinó que fa servir el programari client instal·lat en el seu propi ordinador. Per a comunicar-se, el programa client i el programa servidor es posen d'acord a fer servir un protocol de comunicacions comú, que generalment està pensat per funcionar a la web sobre el protocol HTTP, amb l'objectiu intercanviar fitxers (Badard *et al.*, 2001; Moshfeghi *et al.*, 2004).

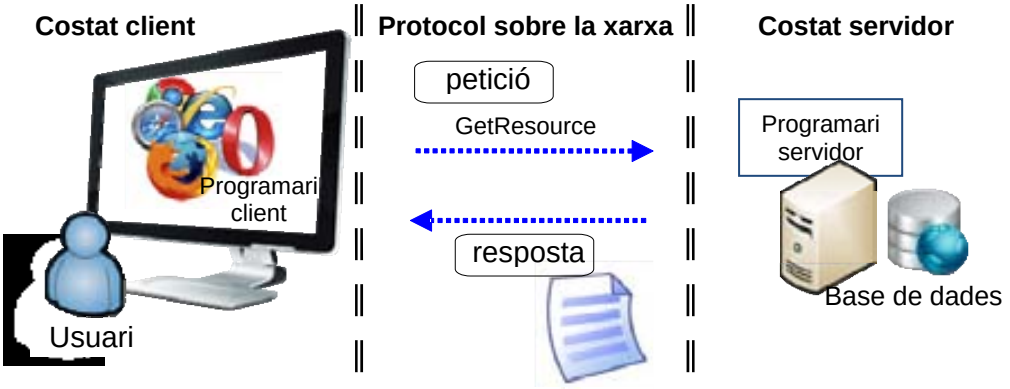


Figura 1: Sistema client - servidor i protocol de comunicacions (font: elaboració pròpia).

1.A.1.3. La interoperabilitat i estàndards per a la distribució de dades espacials

Naturalment, cada responsable d'un conjunt d'informació geogràfica intenta fer la seva feina el millor possible escollint el sistema informàtic, el model de dades i el format que més s'adapta a les seves necessitats. Això fa que diferents actors de diferents comunitats i disciplines adoptin solucions diferents. Quan aquests actors intenten compartir la seva informació amb altres comunitats es fan visibles les diferencies entre aquests sistemes i apareixen les dificultats de compartir informació. Això és el que condueix a una falta d'interoperabilitat entre sistemes, un problema que afecta encara més l'estudi de problemes globals per part de les ciències de la Terra, com per exemple els estudis de canvi climàtic global, on es combina informació multidisciplinar produïda per agencies de molts llocs diferents del món.

Com hem explicat abans, el descobriment de noves i velles fonts d'informació rellevant s'aconsegueix a través de protocols de catàlegs basats en estàndards i metadades. Molts treballs i estudis encara necessiten invertir molt de temps només per a descobrir i reunir els conjunts de dades i les eines necessàries per a tractar-les. El World Wide Web (WWW) i els motors de cerca d'Internet han reduït dràsticament el temps necessari per a trobar informació textual. Si volem construir un sistema similar per a

dades geogràfiques, només serà possible a partir de l'adopció de serveis web implementats a partir de protocols comuns i d'acords d'interoperabilitat (Percivall, 2010).

Una primera definició d'interoperabilitat diu que un sistema és interoperable si pot ser accedit per altres sistemes sense necessitat d'un esforç tècnic i humà important[4]. A la pràctica això vol dir que sistemes fabricats per proveïdors diferents poden ser interconnectats amb uns mínims ajustos. Bàsicament, la interoperabilitat presenta quatre components bàsiques, encara que alguns autors en suggereixen més (Mohammadi 2010, Manso-Callejo 2009, Manso 2009):

- Sintàctica: Permet la reutilització d'informació per a la visualització, la consulta i l'anàlisi. Així, programaris de diferents fabricants poden accedir als mateixos recursos i compartir els resultats.
- Estructural: Refereix a la capacitat de compartir models de dades i de serveis i la capacitat de convertir un model de dades en un altre model de dades.
- Semàntica: És la capacitat del sistema de manipular la informació aplicant regles basades en els conceptes i definicions que usen les estructures de dades fins i tot entre camps de la ciència diferents (Bishr, 1998).
- Corporativa: És la disponibilitat i dificultat que tenen les organitzacions de col·laborar entre elles i compartir la informació.

Les organitzacions d'estandardització treballen per a resoldre els problemes d'interoperabilitat suggerint solucions transversals que poden ser aplicades de manera general. Les principals organitzacions d'estandardització que treballen per a facilitar la interoperabilitat geospacial[5] són:

- El comitè tècnic 211 (TC211) de l'ISO, responsable de la sèrie ISO 19xxx, el més popular dels quals és l'ISO 19115 sobre metadades (ISO 19115:2003).
- L'Open Geospatial Consortium (OGC), responsable de la majoria d'estàndards de serveis de dades geospacials. Una detallada descripció d'aquest organisme i els seus estàndards es pot trobar a l'Annex I.
- El Federal Geographic Data Committee (FGDC), responsable dels estàndards geospacials per a l'administració dels Estats Units d'Amèrica i de l'estàndard de metadades Content Standard for Digital Geospatial Metadata (CSDGM).
- El CEN TC 287, responsable de la transposició europea dels estàndards ISO i de l'adopció dels estàndards rellevants per a INSPIRE.

La principal missió dels organismes d'estandardització és l'edició d'estàndards i recomanacions. Segons el seu propòsit, els estàndards es poden classificar en diverses categories, les més important de les quals són (Bai *et al.*, 2009):

[4] A vegades es confon la interoperabilitat amb la idea del *connectar-i-utilitzar*, en anglès *plug-and-play*, Per a que dos sistemes es considerin interoperables no és imprescindible que els sistemes siguin capaços de dialogar només de connectar-los, encara que sigui desitjable.

[5] En aquesta tesi fem servir el terme *geospacial* quan en referim a serveis i estàndards, per influencia d'alguns autors anglosaxons (i de la lletra G del OGC), però en general em tendit a fer servir *geogràfic* o *espacial* en el seu lloc.

- Serveis de catàleg i registre: Faciliten l'accés a un catàleg de metadades que conté un inventari de dades, de serveis o d'altres recursos. Faciliten el descobriment de dades. L'estàndard de catàleg Catalogue Service (CSW) n'és l'exemple més conegut (Nebert, 2007).
- Serveis d'accés a les dades: Faciliten l'accés a conjunts d'informació geogràfica continguts en arxius digitals. Per exemple, l'estàndard per a distribució de dades de variació continua és el Web Coverage Service (WCS) (Baumann 2010), l'estàndard per a distribució de conjunts d'entitats és el Web Feature Service (WFS) i l'estàndard per accés a observacions de sensors és el Sensor Observation Service (SOS).
- Serveis de representació[6] i visualització: Faciliten una representació visual simplificada (mapa) de les dades per a ser mostrada directament a l'usuari. L'estàndard de mapes per tessel·les Web Map Tile Service (WMTS), que és el capítol 4 d'aquesta tesi, n'és un exemple (Masó *et al.*, 2010a).
- Serveis de transformació i processament de dades: Apliquen regles i algorismes per a manipular les dades geogràfiques. L'estàndard per a processos genèrics sobre informació geogràfica distribuïda, Web Processing Service (WPS) (Schut 2007), i l'estàndard per a processament ràster, Web Coverage Processing Service (WCPS), en són dos exemples.

Els estàndards de serveis mencionats segueixen el sistema distribuït client-servidor. Quan clients i servidors adopten protocols estàndard ja no cal que el client i el servidor hagin estat fabricats per la mateixa companyia ni en el mateix moment. En un cas general, un sol client pot interactuar amb més d'un servidor de diferents proveïdors i fabricants, tot demostrant la seva interoperabilitat sintàctica (figura 2).

Tot i la presència d'aquests estàndards, encara s'observa dificultats i mancances per a la interoperabilitat, ja sigui per problemes en els propis estàndards (Vanmeulebrouk *et al.*, 2009), o per a problemes en les implementacions i programaris actuals (Yang *et al.*, 2005). Quan analitzem les problemàtiques dels serveis geospacials, diferenciar entre el *software* de client i el de servidor ens serà molt útil per entendre com millorar els sistemes en cada nivell.

Aquesta tesi estudia les problemàtiques dels estàndards emprats per a la construcció de les infraestructures de dades espacials per a posteriorment centrar-se en els serveis de representació i visualització i com millorar el seu rendiment gràcies a l'aplicació d'estratègies de tessel·les i l'ús de formats comprimits JPEG2000.

[6] Usem *representació* com a traducció del terme anglès *portrayal*

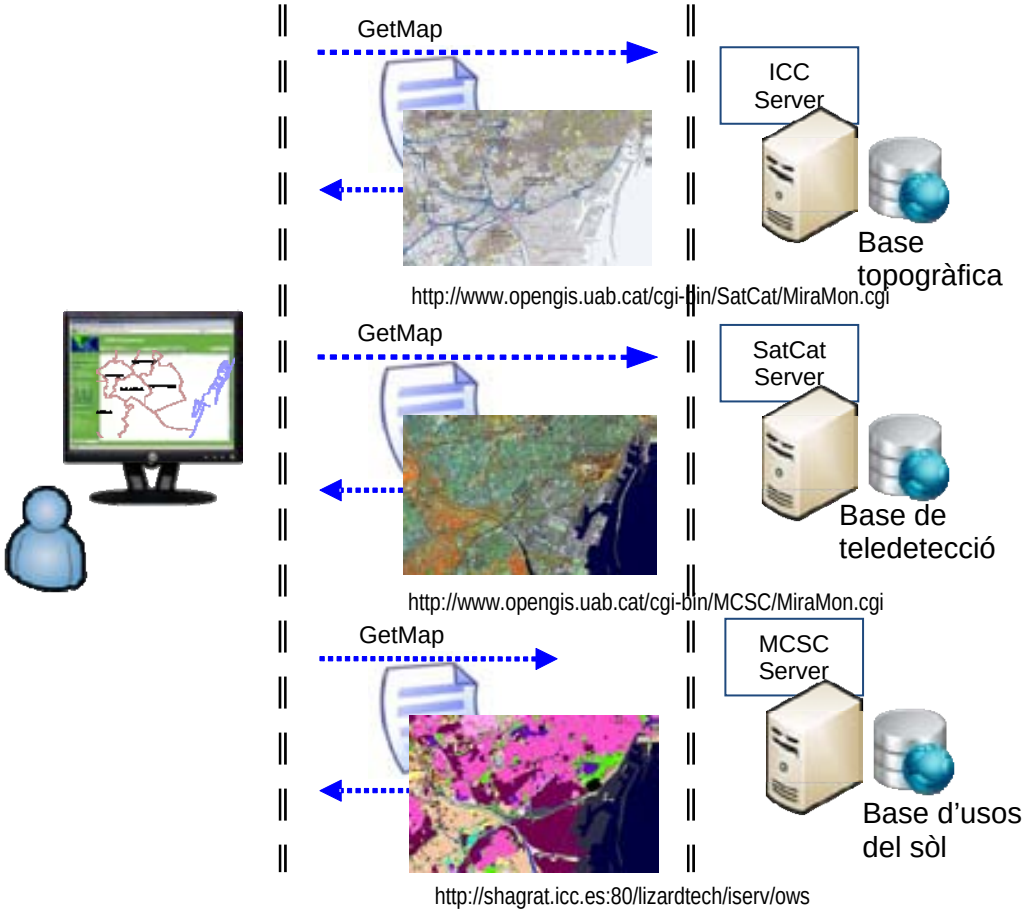


Figura 2: Interoperabilitat d'un programari client amb diversos servidors de mapes (font: elaboració pròpia).

1.A.1.4. Les infraestructures de dades espacials

La definició precisa d'una Infraestructura de Dades Espacials (IDE) ha estat discutida llargament a la literatura (Chan *et al.*, 2001). En aquest treball farem servir una definició simple, que reflecteix les diferents direccions dels elements d'una IDE:

"Una IDE és un sistema integrat que relaciona bases de dades ambientals, socioeconòmiques, institucionals, etc (enllaç horitzontal) i proporciona un flux d'informació des dels nivells locals o nacionals que en alguns casos eventualment arriba a la comunitat global (enllaç vertical)" (Coleman *et al.*, 1997).

Aquesta definició ens diu que les IDE volen integrar tots els actors implicats, per a generar fluxos de dades geogràfiques en totes les direccions i mantenir tothom en el circuit. Una IDE pot ser vista com un magatzem virtual de dades de fàcil accés on els professionals les poden veure, descarregar i treballar (Nebert, 2004), com un mercat on les empreses poden obtenir les dades que necessiten per a generar nous recursos de valor afegit que després són venuts a la infraestructura (van Loenen, 2009), o com un lloc on els usuaris poden generar noves dades col·lectivament, de forma voluntària i contribuir activament al creixement de la infraestructura (Budhathoki *et al.*, 2008). Les

paraules "dades" i "informació" signifiquen aquí "dades geogràfiques digitals", encara que altres tipus de dades també poden ser incloses. D'altra banda, i per simplificar el discurs, no pretenem aquí entrar en la discussió de la frontera entre dades i informació i demanem que el lector tingui una actitud oberta, gairebé sinonímica, entre els dos termes.

En la primera generació de les IDE es va tendir a generar un gran contenidor de dades on es concentrava la informació geogràfica (Alameh, 2003). De seguida es va veure que això generava forts problemes de manteniment i actualització de les dades i que requeria una capacitat d'emmagatzematge massa gran. La segona generació de les IDE se centra en la creació d'una comunitat i en la creació d'un enorme directori actiu que cataloga metadades, dades i actors (figura 3). En aquesta segona aproximació distribuïda, l'adopció estàndards per a la interoperabilitat és clau. La infraestructura de dades espacials europea, promoguda per la directiva INSPIRE, segueix aquesta segona aproximació (Bernard *et al.*, 2005). Malgrat l'existència de dues generacions de les IDE, el seu objectiu basic no ha canviat significativament en els últims 10 anys (Craglia *et al.*, 2008). Aquesta tesi sosté que la raó és que el veritable objectiu de les IDE encara no s'ha aconseguit, ja sigui perquè les solucions disponibles no s'adopten plenament, o perquè s'han implementat de manera incompleta i/o ineficient. Una demostració de la insatisfacció que han causat alguns d'aquests sistemes és que algunes iniciatives recents de compartició de mapes a Internet com http://geocommons.com ha retornat l'esquema de la primera generació de les IDE.

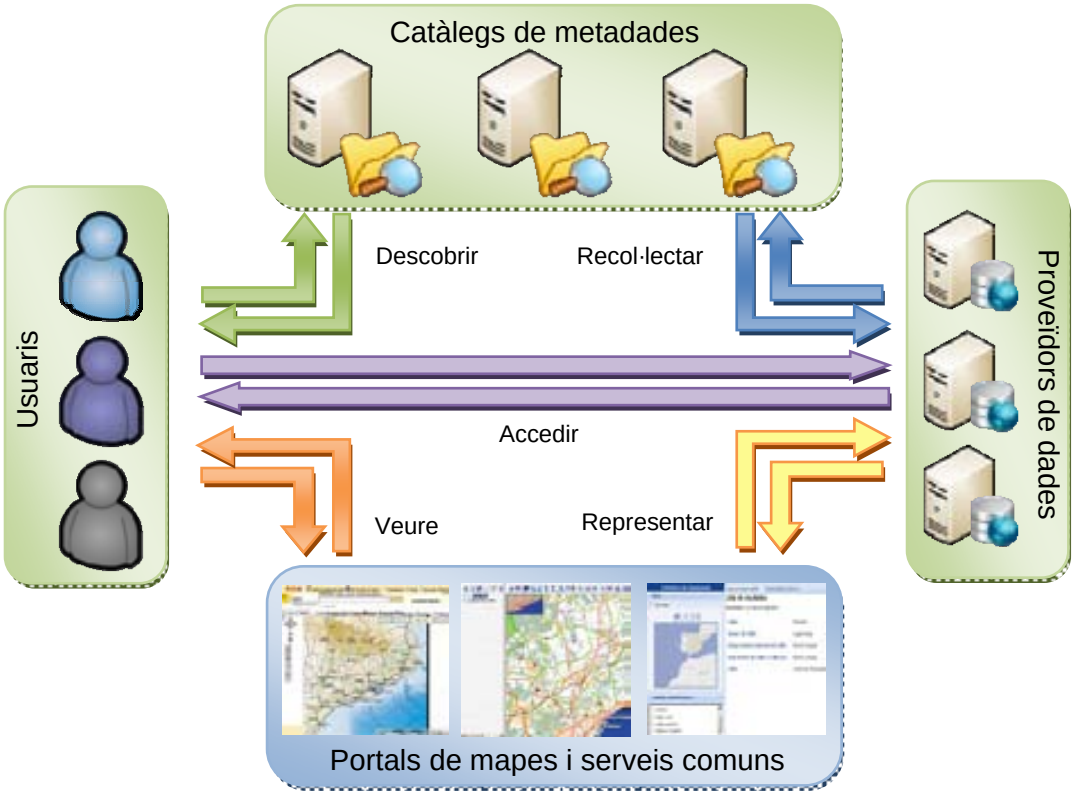


Figura 3: Components bàsics de les IDE de segona generació (font: elaboració pròpia).

El capítol 2 d'aquesta tesi analitza la situació actual i les possibles millores a les IDE.

Les mateixes tecnologies utilitzades en les IDE es poden fer servir en estructures menys estructurades com els sistemes de sistemes (SoS). En ells, diversos sistemes de distribució de dades espacials s'uneixen en un sistema més gran que els engloba. En aquest tipus de sistemes, l'esforç de connexió al SoS ha de ser baix i en base a acords d'interoperabilitat entre els sistemes existents. El Global Earth Observation System of Systems (GEOSS) és un sistema global compost per un conjunt de sistemes d'observació de la terra[7] existents, el que permet detectar duplicitats y fomentar la creació d'iniciatives per cobrir mancances dels sistemes actuals (Butterfield *et al.*, 2008). El sistema usa un únic GEOPortal d'entrada més un sistema de redistribució de dades per satèl·lit de comunicacions anomenat GEONETCast.

1.A.1.5. L'Open Geospatial Consortium (OGC).

L'organització internacional OGC està formada per agències governamentals, universitats, empreses i centres de recerca, i té com a missió promoure l'ús d'estàndards i tecnologies obertes en el context dels sistemes i tecnologies de la informació geogràfica i afins. L'OGC desenvolupa les seves activitats a l'entorn de 3 programes: el programa d'especificació d'estàndards, el programa d'experimentació en interoperabilitat i el programa d'adopció. Dins del programa d'especificació, els grups de treball d'estàndards elaboren, per consens, documents que estandarditzen aspectes independents (com puguin ser els serveis de mapes, els catàlegs de metadades, etc), mentre que els grups de treball temàtics debaten com millorar la d'informació geogràfica interoperabilitat en sectors professionals determinats o per a temes concrets (com poden ser la meteorologia, les ciutats intel·ligents, etc). Malgrat l'interès inicial pels estàndards per a interfícies de programació d'aplicacions (Application Program Interface [API]) en diferents llenguatges (com el Simple Features for SQL) que van ser desenvolupats per l'OGC fa ja més d'una dècada, l'aparició de les IDE i la seva necessitat d'establir plataformes interoperables i distribuïdes a la web ha potenciat l'èxit dels estàndards de serveis web que especifiquin, en la mesura possible, estàndards de codificació i de dades. Els serveis web tenen una estructura client-servidor comuna que fa ús d'un protocol de comunicacions que realitza petites interaccions on el client demana al servidor l'execució remota d'una acció i espera una resposta en forma de fitxer informàtic. Per tal de realitzar la petició, els clients envien missatges codificades a la pròpia adreça web del servei (afegint parelles clau-valor, KVP[8]) (figura 6) o construeixen petits documents XML indicant els paràmetres de la seva petició[9].

[7] Concepte *observació de la Terra* no vol dir només *observació remota* (per teledetecció) sinó que també inclou sensors *in-situ*, etc.

[8] Les parelles clau i valor s'afegeixen darrera de l'adreça del servei a partir de posar un símbol '?' i llavors una seqüència d'una clau, un símbol '=' i el seu valor; una clau, un símbol '=' i el seu valor; etc, que se separen entre elles per un símbol '&'. Un exemple d'aquesta notació es pot veure a la figura 6.

[9] Quan el client o el servidor tenen necessitats especials de seguretat o encriptació, els missatges es poden enviar dins d'un missatge Simple Object Access Protocol (SOAP), que proporciona un sistema que combina el cos del missatges amb mecanismes estàndard d'encriptació i seguretat que es lliuren junts.

Totes aquestes estratègies no han estat introduïdes pas per l'OGC, sinó que els estàndards OGC ens indiquen com són els missatges enviats pels clients i com seran les respostes pel conjunt de serveis geospacials introduït anteriorment i desenvolupat a continuació. A més, tots els serveis de l'OGC comparteixen una petició comuna anomenada GetCapabilities i un mecanisme de negociació de versions. La resposta a aquestes peticions depèn del tipus de servei, però habitualment es tracta d'un document codificat en un dialecte específic d'XML. A part d'estàndards de serveis, l'OGC també desenvolupa estàndards de codificació de dades (per a visualització sobre globus virtuals: KML, per dades georeferenciades vectorials (però no només): GML, per a observacions i mesures de sensors: O&M, per descripció dels propis sensors: SensorML, etc), tot això dins del marc de referència de l'OGC que determina la relació entre els diferents estàndards, tant els abstractes, com els d'implementació, com els de perfils. Això persegueix possibilitar que es puguin prendre millors decisions, per exemple per a prevenir o pal·liar els efectes de catàstrofes, o per planificar millor el desenvolupament social. L'Annex I d'aquesta tesi conté una descripció més detallada de l'OGC, les seves activitats i els seus estàndards.

Per tal de poder impulsar un nou estàndard dins de l'OGC es necessita el suport de tres organitzacions membres de l'OGC per tal de formar un grup de treball i començar el procés d'escriptura del mateix. Un cop passada aquesta fase, s'ha d'exposar públicament l'esborrany durant un mes, recollir els comentaris, incorporar-los i finalment sotmetre el document a votació entre els membres "principals" de l'OGC. A més, durant aquest procés s'ha de demostrar que hi ha 3 implementacions de referència. El capítol 4 d'aquesta tesi presenta l'estàndard WMTS que va ser redactat dins del grup de treball del WMS de l'OGC en el programa interoperabilitat i que ha estat assajat dins de l'OWS-6 del programa d'experimentació de l'OGC, on es varen desenvolupar les 3 implementacions de referència necessàries per a la seva aprovació final, a principis de 2010.

1.A.1.6. Estil d'arquitectura orientat a serveis

El sistema client-servidor utilitza un protocol per a comunicar-se. L'estil d'arquitectura orientat[10] a serveis (SOA) es aquella que fa crides a procediments remots (RPC)[11] (zur Muehlen *et al.*, 2005). Així, podem fer una crida a un servei de mapes perquè generi un mapa personalitzat, o podem fer un crida a un servidor d'entitats vectorials perquè ens generi un llista d'objectes que compleixen unes determinades condicions. En principi, la majoria d'estàndards de serveis de l'OGC segueixen aquesta orientació. Conceptualment, aquesta arquitectura dóna més protagonisme a l'acció remota que s'executa (seguint amb els exemples anteriors les peticions GetMap i GetFeature, respectivament) que no pas a les entitats que es vol recuperar. Això ha conduit a separar les accions en diversos estàndards, agrupant-les pels tipus d'entitats sobre les quals treballaran (seguint amb els exemples anteriors WMS i WFS respectivament). A

[10] L'OGC va introduir l'expressió *service oriented architectural style* per tal de fugir de determinats conceptes establerts anteriorment que resultaven confosos o lligats a marques comercials. En aquesta tesi ho traduïm com estil d'arquitectura orientat a servei, on el terme "orientat" està masculí perquè entenem que "l'estil" és "orientat" i no "l'arquitectura"

[11] En aquest text usarem *crides a procediments remots* (RPC) i estil d'*arquitectura orientat a serveis* (SOA) com a sinònims, però en realitat SOA és un concepte més general que RPC.

la pràctica, però, moltes vegades els servidors de metadades no connecten bé amb els serveis de visualització ni amb els serveis de descarrega, cosa que dificulta que l'usuari pugui passar de manera transparent de la fase de descobriment a la fase d'avaluació i accés a la informació. Un altre dels inconvenients d'aquesta arquitectura és que les dades queden ocultes darrera els serveis, moltes vegades no són accessibles o ho són només de manera parcial (Kim 2004).

1.A.1.7. Estil d'arquitectura orientada a recurs

Els hipervincles tradicionals a la web han donat lloc a l'estil d'arquitectura orientat a recurs, que està guanyant adeptes en el món geogràfic perquè proveeix un nombre limitat d'operacions comunes que poden ser aplicades a recursos diferents, el que s'anomena interfície uniforme. Cada recurs rep un identificador únic, el que facilita el seu intercanvi i la seva actualització. Encara més, els recursos es relacionen entre ells a través de vincles. Els recursos no es poden recuperar perquè són idees abstractes però sí les seves representacions (que estan associades a un format informàtic de dades).

Un tipus d'arquitectura orientada a recurs és el REpresentational State Transfer (REST) (Fielding, 2000). Aquest és l'estil d'arquitectura que segueix la web i l'HTTP. En ella, hi ha bàsicament 4 operacions (anomenades també mètodes, o verbs): (GET=obtenir; POST=crear; PUT=actualitzar; DELETE=esborrar) (Mazzetti *et al.*, 2009). El que resulta més interessant del REST és que els recursos que estan relacionats de manera natural en el model reben identificadors que segueixen un patró de navegació comú.

1.A.1.8. El SIG MiraMon

Des de l'aparició dels primers paquets de SIG de propòsit general als anys vuitanta (Antenucci *et al.*, 1991), son molts el fabricants que han creat el seu propi programa. A mitjans de la dècada dels noranta apareix la necessitat d'integrar aquestes plataformes en el sistema operatiu Windows per tal de poder realitzar desenvolupaments independents de la targeta de vídeo i de la impressora. És en aquest context que, al 1994, es va començar a desenvolupar el que seria el MiraMon. El MiraMon (Pons, 2002) és un Sistema d'Informació Geogràfica i software de Teledetecció que permet la visualització, consulta, edició i anàlisi de dades ràster (imatges de teledetecció, ortofotos, models digitals del terreny, mapes temàtics convencionals amb estructura ràster, etc) i vectorials (mapes topogràfics i temàtics que contenen punts, línies o polígons) i que també connecta als serveis OGC (per exemple, WMS i WMTS) i a fonts de dades geogràfiques organitzades en formats tabulars, per exemple en el context de bases de dades convencionals més o menys adaptades a les necessitats de la informació geogràfica. Els atributs no espacials s'emmagatzemen en taules de bases de dades relacionals. El MiraMon està desenvolupat per diversos membres de les GRUMETS, un grup de recerca consolidat format per membres del de la Universitat Autònoma de Barcelona (UAB), el Centre de Recerca Ecològica i Aplicacions Forestals (CREAF) i l'Estación Biológica de Doñana (CSIC). El MiraMon és principalment un programari d'escriptori que corre sobre sistemes operatius Windows desenvolupat en llenguatge C, tot i que també existeixen parts destinades a ser servidors o clients de dades a Internet, etc, que esmentarem tot seguit.

D'altra banda, el juny de 1993 apareix el que es considera un dels primers prototipus de servei de mapes per Internet, el Xerox PARC Map Viewer. El 1997 (Doyle,1997) començaren les activitats de l'OpenGIS Consortium[12], que organitzaria el seu primer *testbed* anomenat Web Mapping Testbed, i que culminaria amb la publicació de la primera versió de l'OpenGIS WMS interface a l'abril de l'any 2000. El 2001 el MiraMon obtingué un projecte en una convocatòria competitiva de comunicacions avançades per a la Internet de segona generació a Catalunya, el que acabaria essent la llavor del MiraMon Map Server (MMS), un component d'aquest programari que proporciona els serveis web d'acord amb els protocols i estàndards establerts per l'OGC, com ara WMS, WMTS, WCS i WFS. L'aplicació de servidor és un executable de tipus CGI[13], desenvolupat en llenguatge C, que pot ser instal·lat directament a un servidor web del Windows (Internet Information Server, Apache, etc.). Al mateix temps que el servidor, es desenvolupa també el Navegador de Mapes del MiraMon, un client JavaScript[14] que fa d'interfície lleugera per a la consulta de servidors de mapes WMS i WMTS, i que també proporciona serveis descàrrega d'informació seguint el protocol WCS. El Navegador també suporta WFS i WFS-T (limitat de moment a entitats de tipus punt).

El grup de desenvolupament del MiraMon manté l'herència d'aquest gairebé 18 anys de programació i experiència, el que li permet fer evolucionar constantment al programa. Això ha estat particularment útil en el cas de la implementació d'estàndards internacionals que evolucionen al llarg del temps. Durant la recerca continguda en aquesta tesi s'ha realitzat nombroses implementacions de clients i servidors estàndard, feina que no hauria estat possible sense disposar dels fonaments i experiència previs. Així, s'ha realitzat implementacions del WMTS (capítol 4) en les plataformes descrites anteriorment i s'ha participat en l'experiment d'interoperabilitat OWS-6 de l'OGC per a demostrar la interoperabilitat de 3 servidors i 2 clients WMTS. D'altra banda, a la secció 1.A.2.1.2 s'introduirà un portal per a documentar les contribucions dels usuaris finals que es troba descrit amb més detall al final del capítol 2, mentre que a la secció 1.A.1.7 s'introduirà una implementació de l'hipermapa mundial que es troba descrita al final del capítol 3. Ambdues implementacions també han estat realitzades amb tecnologia MiraMon.

1.A.2. Motivació de la tesi

Les infraestructures de dades espacial són una excel·lent plataforma d'implementació dels estàndards per a dades espacials. La realització d'aquesta tesi ve motivada per l'experiència del grup GRUMETS en el desenvolupament i la implantació de tecnologies

[12] Més endavant l'organització canviaria el seu nom a l'actual Open Geospatial Consortium.

[13] Les CGI són una de les aproximacions més antigues i transversals per a desenvolupar aplicacions servidores. Malgrat que els servidors d'Internet per Windows suporten altres tecnologies considerades més eficients (com ara les ASAPI i el .NET), els desenvolupaments resulten massa dependents de la plataforma i el nostre grup de recerca ha evitat el seu ús.

[14] El JavaScript és un llenguatge de programació interpretat que pot ser incrustat dins de planes web per a respondre a les accions que fa l'usuari sense necessitat d'intervenció del servidor. Encara que la seva utilització més comuna és per a controlar el contingut que els usuaris introdueixen en formularis web, permet realitzar aplicacions sofisticades com ara navegadors web de mapes. Inicialment va ser impulsat per Netscape, però el seu ús s'ha generalitzat a tots els navegadors de web habituals. Encara que el nom sembli suggerir-ho, el JavaScript no té res a veure amb el llenguatge Java encara que la sintaxi se sembla vagament.

per a la distribució de dades per a la Internet, com el format MMZ (Pons, 2000) i el servidor i el navegador de mapes del MiraMon, per l'experiència acumulada en la implementació d'estàndards geospacials en el context de les infraestructures de dades espacials (i en especial a la de Catalunya, IDEC) i també pel treball realitzat en els passats 4 anys dins els grups de treball de l'OGC, així com, més recentment, en l'Standards and Interoperability Forum (SIF) del GEOSS. Durant aquests darrers anys, doncs, hem anat detectant diversos problemes en els estàndards actualment utilitzats pel desenvolupament de les IDE, tant a nivell general com més concretament en els serveis de mapes. Aquesta tesi vol constatar i situar aquestes problemàtiques i aportar-hi discussions teòriques crítiques, mesures de rendiment i aplicar solucions.

1.A.2.1. Problemàtiques conceptuals i d'arquitectura derivades de la implementació d'estàndards geospacials en infraestructures de dades distribuïdes.

La secció titulada "Use case 1: accessibility to healthcare centre" del capítol 2 d'aqueta tesi (Masó *et al.*, 2012a), planteja un cas d'ús relativament ingenu on es vol fer servir la Infraestructura de Dades Espacial de Catalunya (IDEC) per localitzar els recursos necessaris per a fer un estudi d'accessibilitat de la població als centres de salut (Olivet *et al.*, 2008). Aquest cas d'ús revela que:

- A part d'eines de cerca i visualització de cartografia, fan falta altres eines de processament de dades basades en estàndards reconeguts, com el WPS, però que no se'n troben.
- Alguns serveis estan disponibles com a portals pels usuaris web, però no hi ha una interfície per utilitzar els serveis des d'altres serveis en un flux automàtic de treball (Masó *et al.*, 2011b).
- No podem retornar informació dels nostres problemes amb la cerca de dades al sistema per sol·licitar correccions o fer-hi aportacions.
- No hi ha una manera fàcil d'integrar els resultats i dades del nostre estudi, de tornada, a la IDE.
- En moltes ocasions, si coneixes els actors involucrats en la producció cartogràfica del país, obtens millores resultats consultant-los directament en comptes de fer servir el propi catàleg de la IDE. A més, sovint al catàleg li falten actors o productes cartogràfics importants per a poder fer l'estudi desitjat, malgrat que realment existeixen.

Aquestes observacions no són exclusives de la IDEC, i és segur que moltes altres IDE presenten problemes similars.

1.A.2.1.1. Situació actual de les infraestructures de dades espacials

A la vista de la situació actual, sembla que l'esquema conceptual adoptat per les IDE és el correcte, però que cal una forta revisió de tots els aspectes de la seva implementació per tal de suggerir i incorporar millores en els aspectes deficients identificats. Aquesta és la principal motivació del capítol 2 d'aquesta tesi. Mentre les millores no arriben, els propis usuaris finals podrien ajudar a mitigar la situació,

contribuint amb el seu propi coneixement i experiència sobre les dades tal i com s'explica a la secció següent.

L'estil d'arquitectura actualment adoptada per les IDE i recomanada per l'estàndard per a les IDE del CEN (CEN 15149) és l'estil orientat a servei. A la secció 1.A.1.61.A.1.7 es proposa l'ús de l'estil d'arquitectura orientat a recurs, que manté força aspectes de l'actual però presenta algunes virtuts, que si s'adoptessin, podrien ajudar a corregir algunes de les problemàtiques actuals.

1.A.2.1.2. La opinió i contribució de l'usuari final (user feedback).

Degut al caràcter institucional de les infraestructures de dades espacials, un dels aspectes que menys s'ha treballant és la contribució de l'usuari final. Budhathoki *et al.* (2008) and Paudyal *et al.* (2009) pensen que la contribució del voluntariat d'informació geogràfica (Volunteered Geographic Information [VGI]) i les iniciatives web 2.0 podrien ser la principal característica de la tercera generació de les IDE.

De fet, la creació d'informació per voluntariat no és una cosa nova. El voluntariat ha ajudat tradicionalment en sectors com l'ornitologia o la meteorologia a crear o enriquir la informació de què es disposa sobre determinades disciplines. Les noves tecnologies han ajudat extraordinàriament a simplificar els mecanismes de centralització d'aquesta informació. Són exemples més recents, la creació del conjunt d'informació de carrers i vies de comunicació públic OpenStreetMap (Haklay *et al.*,2008) i el Weather Observation Website de l'oficina meteorològica britànica (Morris, 2012) (figura 4b).

Els portals de les IDE actuals se centren majoritàriament a oferir eines de consulta a catàlegs de metadades centralitzats sobre dades (que són mantinguts pels diferents centre de suport a la infraestructura) i eines de visualització de dades descentralitzats (que usen els serveis de visualització dels proveïdors d'informació). Seguint l'esperit de la segona generació de les IDE, no es disposa d'un servei de descàrrega d'informació central. La teoria diu que els catàlegs de metadades de dades i serveis poden aportar informació suficient per a descobrir la informació necessària, per a avaluar-la, per a escollir entre la informació trobada (incloent la descripció del model de dades i la qualitat de la informació), per finalment accedir i obtenir la informació original del productor. A la pràctica, per tal de baixar el llistó de les condicions necessàries perquè un proveïdor pugui formar part de la infraestructura, només una informació molt senzilla per a permetre el descobriment de la informació és obligatòria[15] i, per exemple, disposar d'un sistema d'accés a la informació que les metadades representen (ja sigui a partir d'una URL o d'un geoservei d'accés) no és un requisit, i encara menys proporcionar una descripció del model de dades[16]. A més, les IDE rarament incorporen un mecanisme per assegurar que les metadades contingudes en el catàleg representen realment la totalitat dels conjunts de dades disponibles a les institucions que formen part de la infraestructura. Moltes vegades això condueix a una triple frustració per part dels usuaris, que veuen com els resultats de les cerques als catàlegs de metadades no

[15] Aquesta informació bàsica es coneix com el Core metadata, que està compost per 11 elements de metadades dels 409 elements que la ISO19115 defineix.

[16] En realitat el model de dades no es descriu en la ISO19115, sinó a la 19110, que ha estat llargament ignorada.

es corresponen amb la totalitat de la informació disponible, que la descripció sobre la informació no els resulta suficient per a avaluar quin producte els és útil i que la informació d'accés no hi és o no resulta suficient. Aquest problemes són presents tant a les infraestructures de nivell regional com a les infraestructures globals com el GEOSS (Díaz *et al.*, 2012).

Les possibilitats del VGI per a contribuir en l'enriquiment de les IDE han estat identificades per alguns autors però són pocs els treballs que realment presenten implementacions pràctiques d'aquesta aproximació (Goodchild, 2007). Una de les més rellevant s'anomena GeoWiki (Fritz *et al.*, 2009) on els usuaris poden realitzar el control de qualitat de diferents mapes de cobertes i usos del sol a partir de la seva comparació (figura 4a). Seguint estratègies similars, els usuaris podrien contribuir als catàlegs de metadades de les IDE, enriquint-los, actuant com a control de qualitat de les dades, però també aportant nova informació o expressant les seves valoracions. Perquè això sigui possible, calen noves eines de creació de continguts que facilitin la interacció de l'usuari amb el sistema i que li permetin fer les mencionades aportacions que poden ser afegides tant als portals de les IDE com a altres portals relacionats. La secció "Use case 2: Web 2.0 user metadata comments: IDECTalk" del capítol 2 proposa una implementació en aquesta direcció.

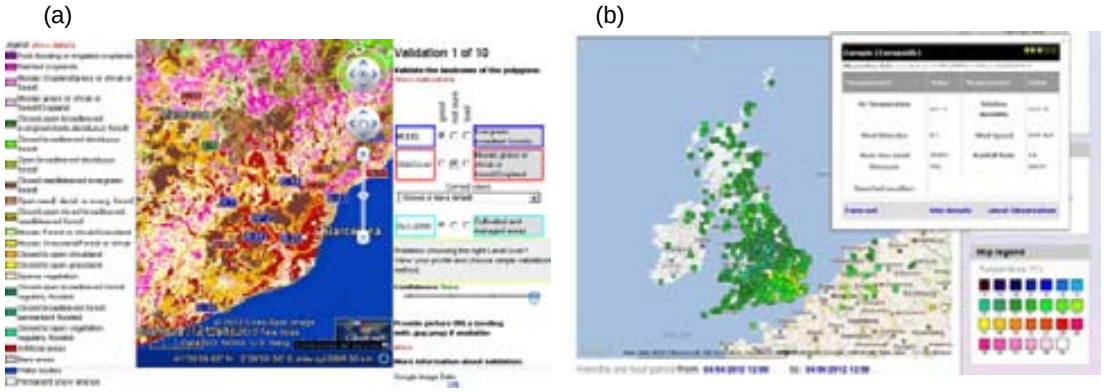


Figura 4: Exemples de portals VGI: (a): GeoWiki (font: http://www.geo-wiki.org/) (b): MetOfficeWOW (font: http://wow.metoffice.gov.uk/).

1.A.2.1.3. L'hipermapa i les seves quatre limitacions

El concepte d'hipertext va ser estès als mapes digitals al 1990 per Laurini *et al.* (1990) en establir el concepte de l'hipermapa. Els hipermapes són sistemes multimèdia georeferenciats que relacionen components individuals els uns amb les altres i amb el mapa (Kraak *et al.*, 1997), de manera que elements del mapa poden ser vinculat a informació de qualsevol tipus (documents de text, imatges, sons, vídeos o altres continguts multimèdia) a través d'hipervincles. Aquests enllaços es representen a la pantalla amb botons que apareixen com a icones o pictogrames, enriquiments tipogràfics en el text, etc (Boursier *et al.*, 1992).

La major limitació de l'hipermapa és que totes les consultes han de ser prèviament definides i "preparades", és a dir, s'ha d'establir vincles entre les entitats, i els usuaris només poden seguir els camins prèviament traçats (Boursier *et al.*, 1992). A més, els

recursos estan vinculats a altres recursos a partir d'identificadors locals i, per tant, hi ha un límit a les seves relacions (Yuan *et al.*, 2000). Una altra crítica és que el model de dades subjacent és extremadament pobre, ja que el concepte d'hipervincles entre les entitats (Boursier et al., 1992; Voisard, 1998) no té una semàntica que identifiqui el propòsit de cada hipervincle. D'altra banda, amb un hipermapa només recuperem recursos (Voisard, 1998), però no som capaços de crear-los, actualitzar-los o esborrar-los.

El concepte de l'hipermapa pot ser estès seguint el mateix patró aplicant el REST als recursos geogràfics d'una manera que aconseguim superar les limitacions de l'hipermapa exposades en el capítol anterior. Així, en una possible implementació REST[17], un recurs és una entitat geogràfica. Un altre recurs és una col·lecció d'entitats geogràfiques que pot tenir diverses representacions en diferents formats (un d'ells pot ser un fitxer GML, un altre un fitxer SHP, etc). Una col·lecció d'entitats està relacionada amb les seves metadades, i pot donar lloc a nous recursos de tipus mapa. El capítol 3 d'aquesta tesi discuteix com es pot aplicar l'estil d'arquitectura orientat a recurs basada en el REST en els SIG distribuïts, tot permetent una millor relació i gestió dels recursos geogràfics.

1.A.2.2. Problemàtiques específiques de la navegació de mapes a Internet i casos d'aplicació

Després d'introduir en la secció anterior els temes que ens han de servir per a discutir les problemàtiques més generals dels estàndards, aquest capítol es concentra en una de les tipologies d'estàndards que hem introduït a la secció 1.A.1.3: els serveis de representació i visualització. Aquests serveis són, sens cap dubte, els serveis més emprats en la cartografia digital perquè mostren la informació d'una manera immediata i visual per a tot tipus d'usuari, ocultant la complexitat de l'estructura de les dades. Es tal l'impacte d'aquests serveis que en molts casos han eclipsat la necessitat de distribuir les dades en el format original, tal i com s'indica al capítol 2.

Els estàndards de representació i visualització es van aplicar originalment a la creació de portals que visualitzaven mapes (representació en forma d'imatge cartogràfica amb la resolució necessària per mostrar al dispositiu de sortida) incrustats en pàgines web. No obstant això, la seva aplicació s'ha estès a tot tipus de productes, des de SIG d'escriptori a aplicacions per a dispositius mòbils.

Els estàndards de representació i visualització (figura 5) són els següents:

[17] Generalment, les implementacions que segueixen l'estil d'arquitectura REST són anomenades implementacions RESTful.

- OGC Web Map Service Standard (WMS) aprovat al 2000 (de la Beaujardiere, 2004), que va ser adoptat més tard com a ISO 19128.
- OGC Web Map Tile Service Standard (WMTS) (capítol 4) aprovat al 2010.
- La ISO 19117 Geographic Information – Portrayal, que defineix els conceptes basics sobre simbolització.
- L'OGC Symbology Encoding (OGC-SE) (Muller, 2006).
- L'OGC Styled Layer Descriptor (SLD) (Muller *et al.*, 2005), que permet aplicar la simbolització SE a un servei WMS .
- L'OGC Geography Markup Language (GML) versió 3, que també pot codificar la simbolització per defecte.

Figura 5: Estàndards de visualització (font: figura 2, capítol 2).

L'estàndard WMS defineix una operació obligatòria (GetMap) per obtenir un mapa d'una zona definida pel seu àmbit i mida en píxels, a partir de les dades d'una o diverses capes (*layers*) (figura 6). Generalment, aquesta representació es codifica en un format típic en els navegadors de web (image / jpeg o image / png). Addicionalment proporciona una operació opcional (GetFeatureInfo) que permet obtenir més informació sobre un punt del mapa, generalment en un format de text, i que s'usa normalment per implementar la consulta per localització (Bermúdez *et al.*, 2012).

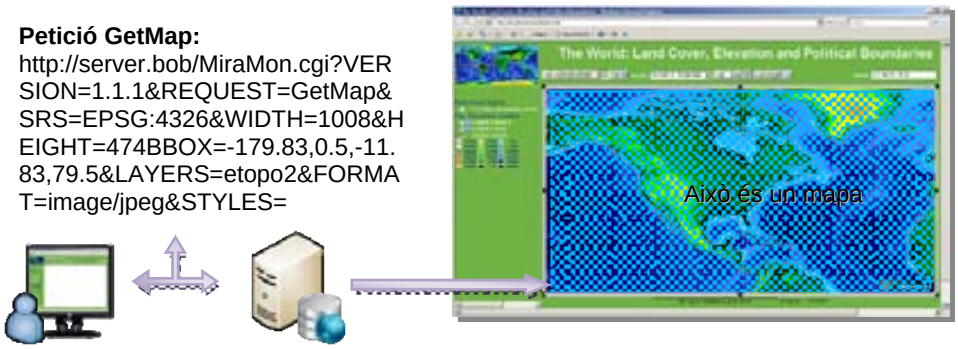


Figura 6: Els clients WMS utilitzen generalment les operacions GetMap per a demanar al servidor tota l'àrea de pantalla de la vista (font: elaboració pròpia).

1.A.2.2.1. Anàlisi rigorosa del rendiment dels servidors de mapes

Un dels principals problemes de moltes implementacions dels servidors de mapes WMS és la seva baixa velocitat de resposta especialment si es demana una escala molt diferent a la de les dades originals, o si hi ha moltes peticions concurrents en el mateix instant de temps (Yang *et al.*, 2005).

És necessari avaluar d'una manera rigorosa i en condicions comparables els diferents productes i estratègies que els fabricants fan servir per tal de millorar el rendiment de les seves implementacions i així determinar les causes de la degradació del rendiment de determinades implementacions en condicions de peticions concurrents.

Una de les causes de la degradació del rendiment dels serveis WMS és més conceptual que tècnica i rau precisament en la flexibilitat del propi estàndard, que fa que difícilment dues peticions siguin exactament iguals. Això impedeix reciclar els resultats de peticions antigues i, encara menys, anticipar les necessitats dels usuaris i realitzar preparacions de vistes.

En aquest sentit, el capítol 6 analitza objectivament el rendiment de diferents productes de diferents fabricants que serveixen mapes emprant protocols WMS, i algunes variants que serveixen tessel·les, en funció del nombre de peticions concurrents i de l'escala sol·licitada.

1.A.2.2.2. Mapes tallats en tessel·les

Una alternativa per augmentar el rendiment dels servidors de mapes és definir un conjunt discret de nivells de zoom (escales) on les peticions seran visibles i alhora definir també un patró de tallat de cada nivell de zoom (Quinn *et al.*, 2010). El servidor només pot respondre a un conjunt de peticions limitat de petits mapes que anomenem tessel·les (*tiles*) (figura 7). Amb aquesta aproximació s'aconsegueix que els diferents mecanismes de *caché*[18] que hi pugui haver, tant en el servidor com en el propi client, com en alguns punt intermedis de la xarxa, puguin treballar més eficientment en tant que existeix una probabilitat més gran que algunes tessel·les siguin demanades diverses vegades. A més, en aquestes circumstàncies també es factible que el servidor tingui completament preparades totes les tessel·les possibles.

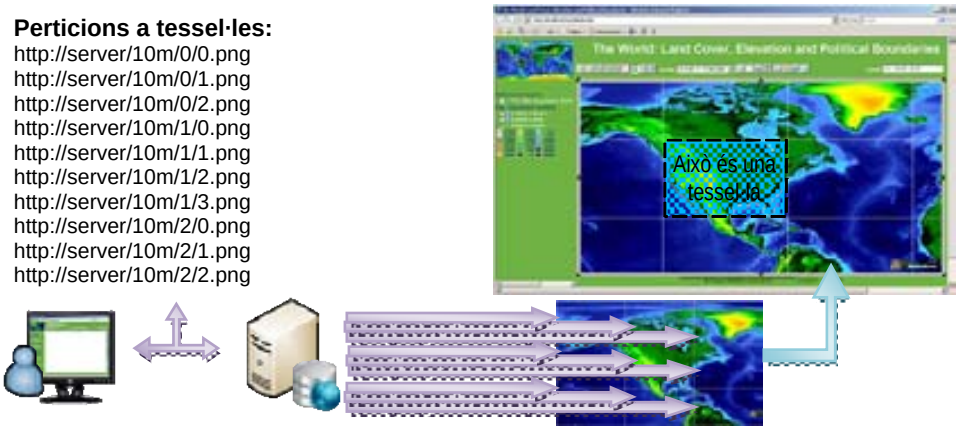


Figura 7: Un client de tessel·les demana peticions simultànies de totes les que cobreixen l'àrea de la vista, excepte en el cas que les tingui en *caché* del client. També oculta les parts de tessel·la que no entren en la vista (font: elaboració pròpia).

[18] En català es pot arribar a traduir com *emmagatzematge de memòria cau*, però preferim utilitzar aquí el terme original.

Aquesta aproximació és la que fan servir productes o serveis com el TileCache (TMS), Google Maps o TerraService (Barclay *et al.*, 2006), i també la de l'estàndard d'OSGEO WMS-C. Cada sistema defineix el seu propi conjunt de nivells de zoom (figura 8b), el patró de tallat de cadascun dels nivells de zoom (figura 8a), la seva pròpia manera d'exposar aquest patró de tallat i la seva pròpia manera de recuperar una tessel·la. L'existència de diferents alternatives dificulta la interoperabilitat, donat que pocs clients poden llegir totes les variants actualment existents.

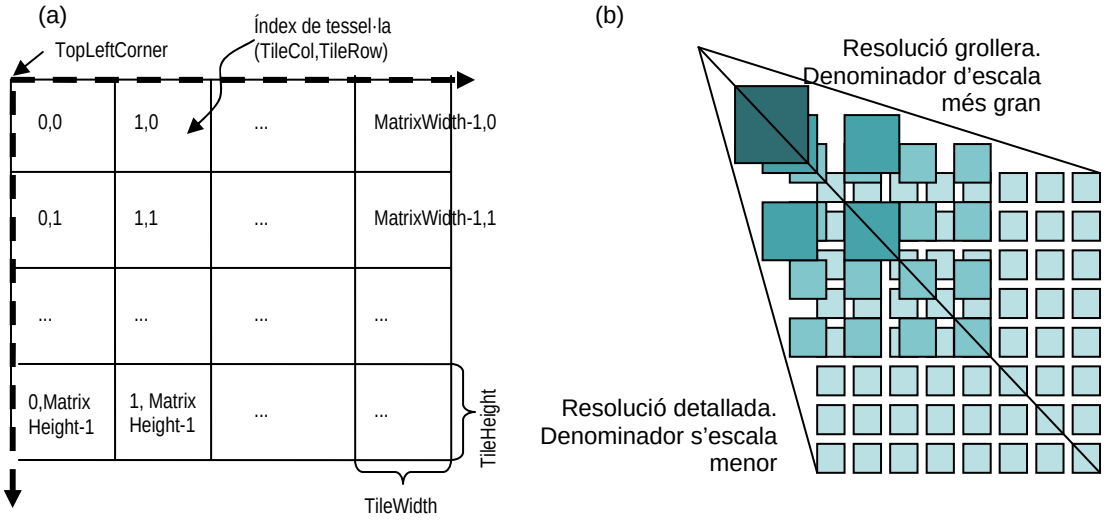


Figura 8: (a) Definició del patró de tall emprat (font: figura 2, capítol 4) i (b) nivells de zoom per a un dels sistemes de mapes en tessel·les, en aquest cas l'estàndard OGC anomenat WMTS (font: figura 3, capítol 4).

Es feia necessària, doncs, la redacció d'un estàndard internacional validat pels principals fabricant de *software* i les principals agencies governamentals. Per això, es va impulsar la creació de l'estàndard WMTS que figura com a capítol 4 d'aquesta tesi. La creació d'un estàndard OGC requereix un procés basat en el consens, tal i com s'ha explicat a la secció 1.A.1.5. El document va resultar aprovat amb 40 vots a favor de les 81 organitzacions amb dret a vot i només 1 vot en contra.

1.A.2.2.3. El format JPEG2000

Una altra alternativa per a la millora del rendiment dels servidors WMS és l'optimització del format intern d'emmagatzematge de les dades i de l'algorisme d'extracció de les mateixes per tal de reduir al màxim el temps de resposta de cada petició d'un mapa.

El format JPEG clàssic és un dels formats més emprats en els servidors de mapes WMS i WMTS donat que pot ser mostrat directament pels navegadors web actuals. El format JPEG2000 ofereix nombrosos avantatges sobre el format JPEG clàssic (Zabala, 2010); la que més influeix en la seva velocitat és que permet accés directe (aleatori) a una part de la imatge. Aquest estàndard internacional (ISO15444-1) pot realitzar, a més, compressió amb pèrdua i sense pèrdua, la qual cosa és un detall que pot ser important en contextos en què preocupi disposar de la màxima qualitat. El JPEG2000 ofereix una millor qualitat en la mateixa relació de compressió (per exemple, no té els artefactes

de blocs de 8x8 presents al JPEG clàssic) i ha estat dissenyat per facilitar la visualització en pantalla d'imatges molt grans (particularment de to continu). Així, permet recuperar imatges a diferents resolucions i mides a partir del mateix fitxer comprimit (el JPEG clàssic només pot recuperar imatges a la resolució fixada i descomprimint tot el fitxer). No obstant això, per obtenir tots aquests beneficis el cost computacional dels algorismes és molt més elevat, el que fa que els temps de compressió (i, segons com, de la descompressió) completa de la imatge JPEG2000 sigui significativament més gran que el del JPEG clàssic. Per tots aquests motius, aquest format resulta indicat quan es necessari treballar amb imatges molt grans però nomes cal accedir a una zona petita cada cop.

Simplificant-ho molt, dins d'una imatge JPEG2000 es pot trobar emmagatzemada la transformada *wavelet* de la imatge original tallada en rajoles rectangulars anomenades *codeblocks*[19] (figura 9). La transformada *wavelet* presenta la virtut que és una transformació reversible i, a més, que és possible obtenir la imatge original amb menor grau de detall obtenint només una part dels *codeblocks.*

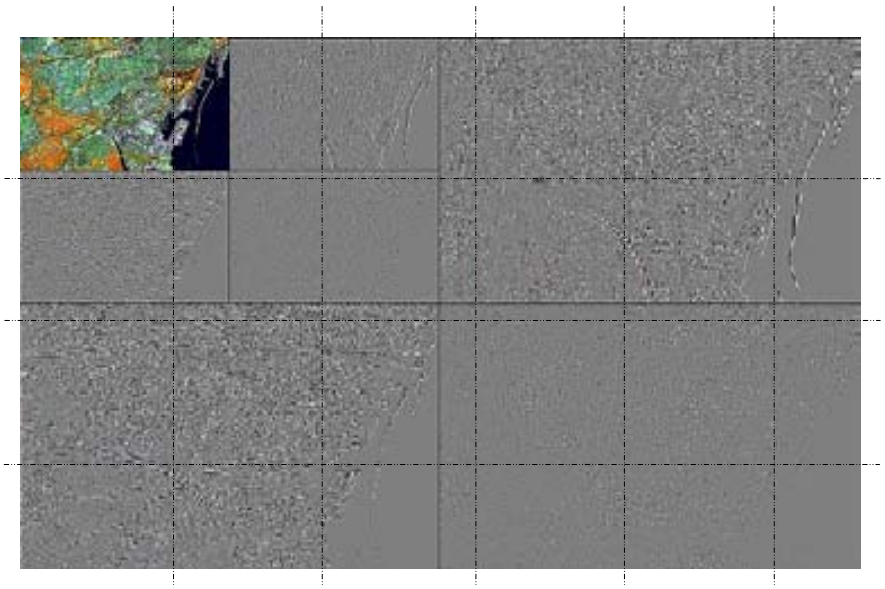


Figura 9: Transformada *wavelet* d'una composició en fals color d'un fragment d'imatge Landsat de l'àrea metropolitana de Barcelona. En línia continua, 3 nivells de resolució de la transformada. Amb línies de punts: divisió en *codeblocks* (només els 2 primers *codeblocks* són necessaris per a recuperar una imatge completa de resolució ¼ de la resolució original) (font: elaboració pròpia amb el programa fwt2d[20]).

Addicionalment, abans de ser transformada i comprimida, la imatge original també pot ser tallada en tessel·les en l'espai no transformat, les quals es comprimeixen de manera independent i es poden emmagatzemar en un sol fitxer JPEG2000 (figura 10).

[19] Resulta fàcil de confondre un *codeblock* amb una *tessel·la*: Un *codeblock* és un fragment rectangular en l'espai transformat *wavelet* mentre que una tessel·la és un fragment rectangular en l'espai original no transformat.

[20] Programa per Windows elaborat per Chesnokov Yuriy el 20 Octubre de 2007. http://www.codeproject.com/Articles/20869/2D-Fast-Wavelet-Transform-Library-for-Image-Proces

A més, l'estàndard ISO15444-2 defineix una manera d'incloure metadades dins de la imatge JPEG2000 com a fitxers XML que és aprofitada per l'estàndard OGC GMLJP2. A més, l'estàndard ISO15444-9 defineix un protocol de comunicacions per a la transmissió incremental d'informació codificada en format JPEG2000. Donades les similituds entre els sistemes de tessel·les i el format JPEG2000, té sentit de plantejar si el JPEG2000 pot ser una alternativa o un complement al WMTS.

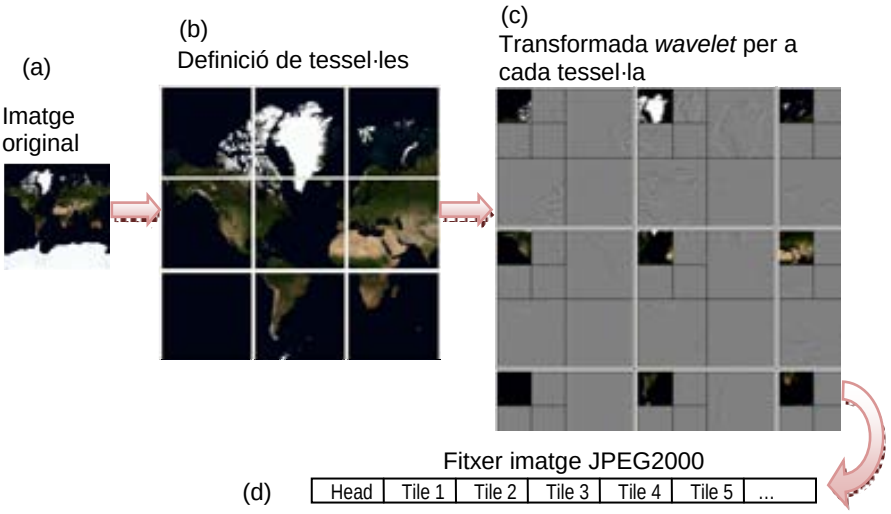


Figura 10: La imatge original (a) es pot dividir en tessel·les (b) que són transformades *wavelet* (c), tallades en *codeblocks* i incorporades a la seqüència de bytes del fitxer JPEG2000 (d) (figura basada en: figura 7, capítol 5).

El capítol 5 exposa les diferents alternatives per a combinar el JPEG2000 amb els diferents estàndards OGC i proposa que el format JPEG2000 és un bon format per emmagatzemar les dades originals en els servidors WMTS si la compressió es fa d'una manera determinada.

1.A.3. Objectius de la tesi

L'objectiu general d'aquesta tesi doctoral és estudiar l'estat actual dels models de dades d'informació geogràfica (incloent dades de teledetecció) a Internet i les implicacions i dificultats de la seva implementació i desplegament pràctic, especialment en el context de les IDE, tot realitzant propostes per tal de corregir algunes de les problemàtiques trobades.

Aquest objectiu general es pot concretar en els següents objectius específics:

El primer objectiu específic és repassar els estàndards emprats en les IDE, el procés de la seva implementació i les dificultats que s'observen en les tecnologies que usen aquests estàndards, proposant millores concretes tant a nivell de servei com a nivell de client que facin evolucionar l'estat de les IDE.

El segon objectiu específic és la definició d'un nou marc tecnològic que permeti reestructurar l'estil d'arquitectura actual emprada en les IDE, basat en una orientació a servei, per una altre estil d'arquitectura orientat a recurs, amb la finalitat de fer un millor ús de la pròpia Internet, relacionar millor els recursos entre ells, així com d'incorporar altres col·lectius, com són per exemple els estàndards usats en polítiques de dades obertes (*open data*), la cartografia generada per voluntariat i els navegadors de mapes del mercat de masses i els globus virtuals.

El tercer objectiu específic és contribuir a la millora del rendiment de les implementacions del visualitzadors de mapes basats en estàndards a partir de proposar estratègies per incorporar el JPEG2000 i de proposar un nou estàndard de servei de mapes en tessel·les anomenat WMTS.

El quart objectiu específic és incorporar els nous estàndards a les tecnologies del programari MiraMon (tant en la versió d'escriptori com en el navegador i servidor de mapes) (Pons, 2002) per tal de validar-les a través del seu ús, procés que ha de permetre dissenyar sistemes que optimitzin els algorismes que implementen les anteriors propostes.

Finalment, el cinquè objectiu específic és analitzar quantitativament els diferents productes de serveis de mapes (incloent les implementacions realitzades en el MiraMon) i valorar fins a quin punt les millores proposades contribueixen a la millora del rendiment de les implementacions.

1.A.4. Organització de la tesi

L'anterior introducció té com objectiu situar al lector en el context general necessari que li permeti llegir els diferents capítols de la tesi; es troba traduïda al anglès en l'apartat 1.B. Cal considerar que en ser aquesta una tesi per compendi de publicacions, cada capítol conté un resum (*abstract*) i la seva pròpia introducció individualitzada, el que permet aprofundir una mica més en els aspectes més concrets tractats per aquest capítol.

La tesi s'estructura en dos grans blocs. El primer està format pels aspectes més generals que descriuen diverses problemàtiques d'aplicació dels estàndards a les IDE i fa propostes concretes per a millorar-lo, tot suggerint una nova arquitectura REST per a crear l'hipermapa mundial. El segon gran bloc exposa aspectes més concrets que fan referència a la millora dels serveis de mapes, a partir d'incorporar el JPEG2000 o d'estratègies basades en tessel·les, així com la mesura del rendiment d'algunes d'aquestes implementacions.

El treball es presenta com a compendi de publicacions i està format per 5 publicacions que es troben en els capítols centrals i que descriuen amb més detall els aspectes introduïts en el present apartat; un capítol final amb el resum de resultats i les conclusions que serveixen per donar una visió conjunta als assoliments. També s'inclou una publicació addicional en l'Annex I, que pel fet de ser actualment en estat de revisió no s'ha pogut incloure al cos principal de la tesi, però que pensem que té interès en el context de la recerca realitzada i complementa adequadament el conjunt.

Els aspectes acabats d'exposar poden ser detallats una mica més, abans d'entrar a fons en cadascun d'ells, amb la següent visió general.

Dins dels aspectes generals dels estàndards geospacials contemplats a la tesi hi trobem:

Com un complement introductori, l'Annex I d'aquesta tesi presenta una revisió dels principals estàndards de l'OGC i els seus fonaments, donant un enfocament més pràctic més enllà de la mera repetició de contingut dels propis estàndards. També s'acompanya d'esquemes originals realitzats expressament. Representa, creiem, una bona introducció als temes més concrets que es desenvolupen en la tesi. La publicació a estat enviada per a formar part del llibre "Fundamentos para las IDE", que publicarà la UPMPress i que serà editat per Miguel A. Bernabé-Poveda i Carlos M. López-Vázquez (Bermúdez *et al.*, 2012)

El capítol 2 de la tesi repassa la generació actual de les IDE i proposa maneres de millorar el seu rendiment. S'ha publicat a la revista *International Journal of Geographical Information Science,* indexada pel JCR-SCI (Masó *et al.*, 2012a). També exposa dos casos d'ús de manera més detallada: l'un sobre l'ús de la IDEC per a un estudi d'accessibilitat als centres de salut, i un altre sobre un entorn perquè els usuaris puguin realitzar comentaris al registres del catàleg de la IDEC.

El capítol 3 d'aquesta tesi suggereix de fer evolucionar el concepte de l'hipermapa cap a l'hipermapa mundial (WWH) a través d'aplicar-hi el nou estil d'arquitectura orientat a recursos REST i que resol alguns dels problemes exposats en el capítol 2; alhora que exposa com implementar-ho en el *software* MiraMon. Ha estat publicat a la revista *International Journal of Digital Earth*, indexada pel JCR-SCI (Masó *et al.*, 2012b). Aquest capítol es complementa amb l'Annex II que dóna una llista completa dels recursos i operacions definides sobre la idea del WWH.

Dins dels aspectes concrets dels servidors de mapes hi trobem:

El capítol 4 d'aquesta tesi és un estàndard internacional que explica un protocol per exposar una estructura de tessel·les que recull informació geogràfica en forma de mapa (de manera pictòrica), i per a sol·licitar i rebre una tessel·la. Aquesta proposta va ser acceptada per l'OGC el 2010 (Masó *et al.*, 2010a). El seu impacte es pot mesurar a partir de les 16 citacions en treballs científics diversos que s'han produït fins al moment (febrer 2012).

El capítol 5 d'aquesta tesi analitza com es pot millorar el rendiment dels servidors de mapes amb l'aplicació del format JPEG2000. S'ha publicat a la revista *Italian Journal of Remote Sensing (Rivista Italiana di Telerilevamento),* indexada pel JCR-SCI (Masó *et al.*, 2010b).

El capítol 6 d'aquesta tesi fa una anàlisi de l'impacte de les peticions concurrents en les implementacions dels fabricants de servidors de mapes més importants del sector ja sigui en WMS o en WMTS. Aquesta publicació forma part dels *proceedings* del congrés internacional organitzat el 2011 per la International Academy, Research and Industry Association (IARIA) (Masó *et al.*, 2011).

El capítol 7 fa un resum de resultats continguts en els capítols 2 a 6. Finalment, el capítol 8 enumera les conclusions de la tesi. L'Annex III inclou una llista d'acrònims.

1.B. INTRODUCTION (English version)

1.B.1. General introduction

The introduction of the Internet first into universities and then into businesses and at other educative levels, then later into people's homes, has meant that all computers, which before were disaggregated, have now become interconnected, generating a flow of data and services that allows information and user experiences to be exchanged instantaneously in a process called "network connection". This is mainly possible due to email and the Web. Since the beginning of this process, digital map users and producers have recognized the enormous potential for working altogether in a network or for spreading and sharing their information and reducing redundancies (Meeksa *et al.*, 2004). The number of digital map users has grown exponentially because end users can access these maps easily without intermediaries. This process accelerated greatly with the introduction of tools produced by the main Internet search engines (*e.g.*, MapQuest, Michelin, and soon after, Google Maps, Bing Maps, etc.) and virtual globes (Butler, 2006, Sheppard *et al.*, 2004). However, there were some problems, such as the lack of standardization in the data formats, the visualization software with intelligible symbolization, the copyright, the high cost of the maps (usage fees) in some areas and countries, the narrow bandwidth of the exchange of big data, and the lack of user training, etc. (Tu *et al.*, 2004, Yang *et al.*, 2005). Recently, this paradigm change process has become even faster with the appearance of cloud computing, so that the storage and processing capacities are also shifting from the local computer to the network. However, the cloud computing is so new that its impact is still difficult to predict (Yang *et al.*, 2011).

Standards bodies (World Wide Web Consortium [W3C], International Organization for Standardization / Technical Committee 211 [ISO/TC211], Open Geospatial Consortium [OGC], etc) are actively working on setting up specifications to improve the user experience by ensuring interoperability in formats and procedures without removing necessary functionalities (Albrecht, 1999; Percivall, 2010). Moreover, the map agencies are organizing themselves around Spatial Data Infrastructures (SDI), which are organizations that aim to promote the discovery, evaluation, access, distribution and use of data at different geographic levels. In order fulfil these objectives, these organizations advocate, coordinate and stimulate the adoption of standards. Despite the positive aims of these organizations, and their considerable influence in the industry, the standardization process is far from reaching its goals (Crompvoets *et al.*, 2004). To make the situation worse, some independent actors that have recently emerged, such as Google Maps/Earth and GPS systems for the car industry, have chosen not to adopt the standards for their own business interests.

1.B.1.1. The need for digital cartography and its dissemination problems

Digital cartography has many advantages over traditional paper maps (Pons, 1996, Heywood *et al.*, 2006):

- The non-spatial information is stored as original data (usually in tables) and not as a symbolization characteristic of the spatial feature (colour, thickness, etc).
- The way entities are symbolized can easily be changed to emphasize other aspects of the data.
- The information can be transformed (for example by changing the projection or by applying a generalization or a change of scale).
- It is possible to work with individual entities that have a unique identifier (see chapter 3).
- Links between entities can be set up as well as links with external information (see chapter 3).
- A dataset can be kept to date more easily.
- The 2D limitations imposed by the paper can be overcome so it is possible to work with multidimensional models (*e.g.*, 3D weather cartography) or with time evolution (*e.g.*, fleet tracking).
- It is not necessary to cut the maps that are too large or have a high level of detail in sheets (although for tradition reasons or due to limitations in file formats it is also sometimes required in digital cartography).
- Information can be transmitted, duplicated and easily printed.
- Although the traditional cartography can overlay information, analyze distances and express some relationships between entities, digital cartography can enormously increases enormously the number of analytical operations that can be applied and also automates analysis of large data volumes. Thus, alternative hypotheses can be tested more easily with repeatable scientific criteria.
- Digital maps do not wear out with its use.

To use digital cartography we need a computerized system that makes it possible to view, query and exploit them. These computerized systems have changed over time, from desktop computers that incorporate GIS software, to mobile devices with navigation systems and map browsing systems for Internet portals. Thus, digital cartography is used in many formats and systems. Indeed, the first problems arise when it is necessary to exchange information. This problem becomes more apparent when these systems are interconnected by communication networks and need to use data from different sources. Thus, data services emerged to facilitate the exchange of information.

Digital cartography groups the information into datasets that basically use two data models: A data model designed to describe variables that cover the area continuously, in which these values are collected in a quasi-continuous pattern (*e.g.*, temperature, elevation, etc.); and a data model designed to identify entities on the world, which can be described individually (*e.g.*, administrative units, buildings, roads, etc.). The first model is called *coverages* and will result in coverages distribution services (coverage services), while the second model is called *features* and will result in feature distribution services (feature services)[21]. These models are employed to store and

[21] To make things simpler, we can say that the coverage model commonly uses a raster representation (where the entities are cells that are more or less regularly distributed in the space and normally contain values of a measured magnitude) and the feature model normally uses a vector representation (where

process the information, but are not directly visualized. To visualize them it would be necessary to simplify them, assign them to a set of symbolization rules that provide colour, texture, small text, etc. to the values or the entities, and adapt them to be displayed on a device (such as a screen or printer) in order to generate a map. This last procedure, in a distributed environment, can be delegated to map services. When these services are available on the Web, they are called Web Coverage Service (WCS) (Baumann, 2010), Web Feature Service (WFS) (Vretanos, 2010) or Web Map Service (WMS) (the Beaujardiere, 2004) (or Web Map Tile Service [WMTS] [Maso *et al.*, 2010]) respectively.

Each dataset represents a specific theme (*e.g.*, hydrography, buildings, humidity, population density, etc). A computer system can store a very large number of datasets, and so the datasets need to be ordered in terms of subject, scale, time, etc. This classification is more or less arbitrary and each agency uses its own approach. Therefore, when different information data stores are joined together, this classification is no longer possible and automatic data searches become necessary. These systems are called metadata catalogues and use structured short descriptions in the form of records for each dataset that lead to the catalogue services. These catalogues make it possible to find the data in the system by using small queries in more or less natural language. When these services are provided through the Web they are called Catalogue Service Web (CSW) (Nebert, 2007).

This PhD thesis explores possible ways of improving the different services and standards that are applied to disseminate digital cartography.

1.B.1.2. The client-server system

The client-server separation is one of the foundations of distributed computing. In this model, the computer system is separated into two independent software units that are usually in different computers (Peng et al., 2003) (Figure 1). The client program makes a request to the remote server, the server program then does the job that the client has requested and sends the result back to the client. The user of the system does not usually interact directly with the server but operates the client software installed in their computer. For dialogue to be possible, the client program and server agree to use a common communication protocol, which is generally designed to operate on the Web with the HTTP protocol, in order to exchange files (Badard *et al.*, 2001; Moshfeghi *et al.*, 2004).

---

the entities are usually points, lines and polygons linked to lists of non-spatial attributes), but this is not always the case.

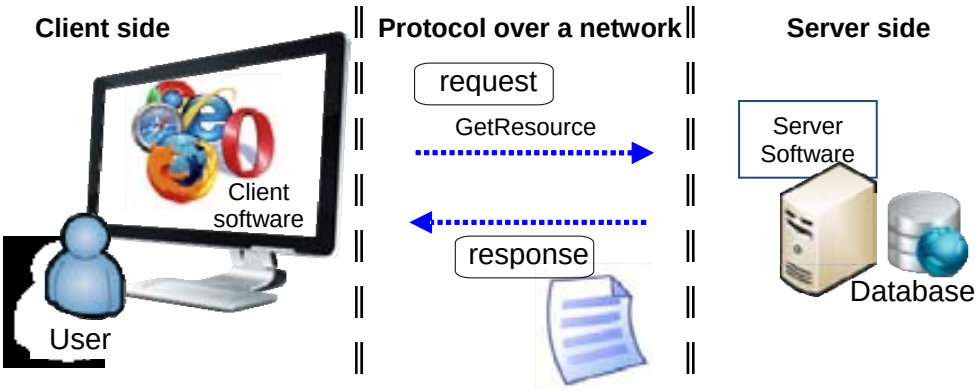


Figure 1: Client-server system and the communication protocol (source: own preparation).

1.B.1.3. Interoperability and the standards for distributing spatial data

Obviously, each dataset responsible tries to choose the most appropriate computer system, data model and format for its needs. This means that different actors from different disciplines and communities adopt different solutions. When these actors try to share their information with other communities, differences between the systems become visible and difficulties arise in sharing information. This means the systems are not interoperable. This problem affects many studies in the Earth sciences and on global issues, such as global climate change studies, which combine multidisciplinary information produced by agencies from many different parts of the world.

As explained above, new and old sources of relevant information are discovered through catalogue protocols based on standards and metadata. Many studies still need to invest a long time simply to discover and collect datasets and the tools necessary for treating them. The World Wide Web (WWW) and Internet search engines have drastically reduced the time needed to find textual information. If we want to build a similar system for geographic data, it will only be possible by adopting of Web services implemented using common protocols and interoperability agreements (Percivall, 2010).

An definition of interoperability says that a system is interoperable if it can be accessed by other systems without significant human and technical interaction[22]. In practice this means that systems manufactured by different vendors can be interconnected with minimal adjustments. Basically, interoperability has four main aspects, although some authors suggest more (Mohammadi 2010, Manso-Callejo 2009, Manso 2009):

- Syntactic interoperability: Allows the reuse of information for displaying, querying and analysing. Thus, software from different vendors can access the same resources and share the results.

[22] There is some confusion between interoperability and the plug-and-play idea. To consider two systems as interoperable, it is not essential that these systems are able to talk to each other by simply connecting them, although it is even better if it is possible.

- Structural interoperability: The ability to share data and service models and the ability to convert a data model into another data model.
- Semantic interoperability: The system's ability to manipulate information by applying rules based on the concepts and definitions that data structures use even across different science domains (Bishr, 1998).
- Corporative interoperability: The availability and difficulty that organizations face to collaborate with each other and share information.

The standardization organizations are working to resolve interoperability problems by suggesting solutions that could be applied generically. The main standardization organizations working to facilitate geospatial[23] interoperability are:

- The Technical Committee 211 (TC211) of the ISO, which is responsible for the ISO 19xxx series, of which the ISO 19115 about metadata (ISO 19115:2003) is the most popular.
- The Open Geospatial Consortium (OGC), which is responsible for the majority of standards for geospatial data services. A detailed description of this organization and its standards can be found in the Annex I.
- The Federal Geographic Data Committee (FGDC), which is responsible for geospatial standards for the United States administration and the Content Standard for the Digital Geospatial Metadata (CSDGM) standard.
- The CEN TC 287, which is responsible for transposing ISO standards to European standards and the adoption of ISO standards relevant to INSPIRE.

The main aim of standardization bodies is to publish standards and recommendations. Standards can be classified into several categories according to their purpose. The most important ones are (Bai et al., 2009):

- Registry and catalogue services: These facilitate access to a metadata catalogue that contains an inventory of data, services or other resources. They facilitate data discovery. The Catalogue Service Web (CSW) (Nebert, 2007) is the best known example of this type of standards.
- Data access services: These facilitate access to datasets contained in digital files. For example, the standard for distributing continuous variation data is called the Web Coverage Service (WCS) (Baumann 2010), the standard for distributing of feature collections is called the Web Feature Service (WFS) and the standard for accessing sensor observations is called the Sensor Observation Service (SOS).
- Portrayal and viewing services: These provide a simplified visual representation of data (a map) to be displayed directly to the user. An example is the map standard for tiled maps called the Web Map Tile Service (WMTS), which is included as the chapter 4 of this PhD thesis (Maso *et al.*, 2010a).
- Data transformation and processing services: These apply rules and algorithms for manipulating geographic data. The standard for generic processes on

[23] In this PhD thesis we use the term *geospatial* when referring to some standards and services accordance with certain (and the letter G of the OGC), but in general we use *geographic* or *spatial* instead.

distributed geographic information, the Web **Processing** Service (WPS) (Schut 2007), and the standard for raster processing, the Web Coverage Processing Service (WCPS), are two examples.

The service standards mentioned above follow the client-server distributed system. When clients and servers adopt standard protocols, it is no longer necessary for implementations to be manufactured by the same company or at the same time. In a general case, one client can interact with more than one server from different suppliers and manufacturers, which demonstrates their syntactic interoperability (Figure 2).

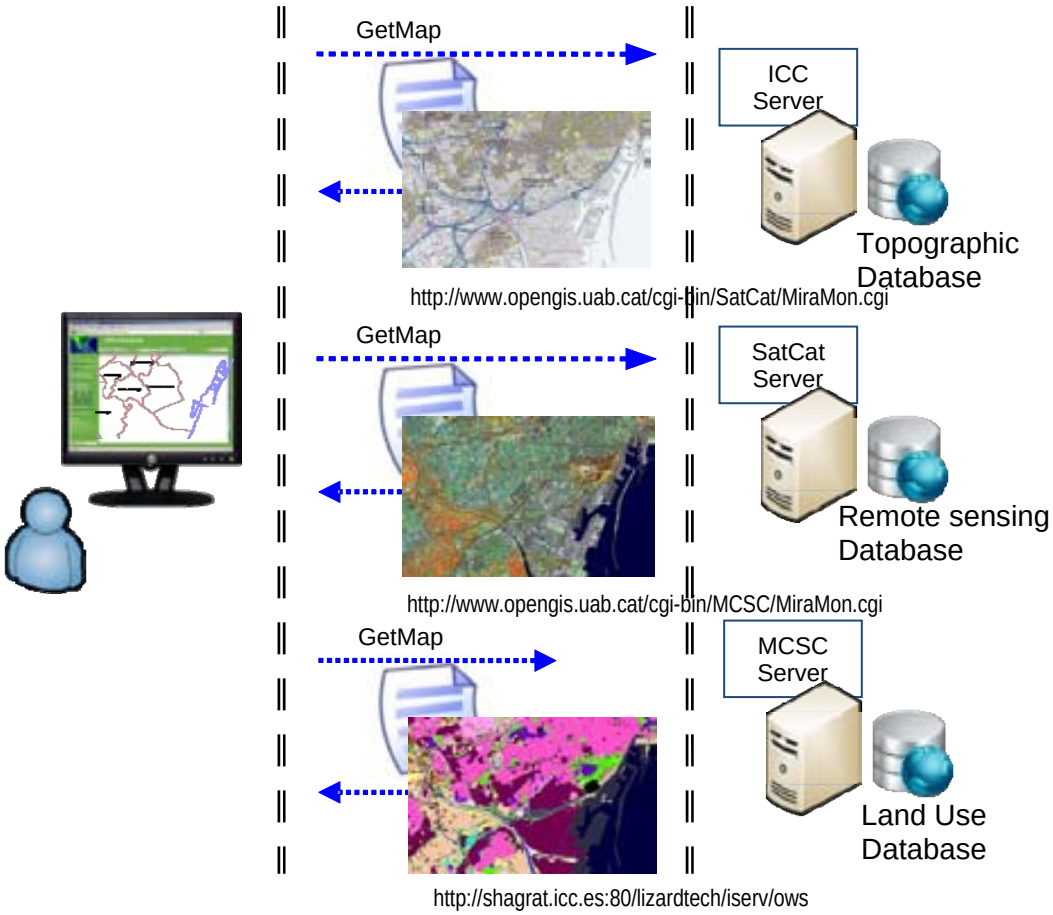


Figure 2: Interoperability of a client software with multiple map servers (source: own preparation).

Despite the standards, there are still some shortcomings and difficulties involved in obtaining interoperability, either due to problems in the standards themselves (Vanmeulebrouk *et al.*, 2009) or problems in current implementations and software (Yang *et al.*, 2005). When the problems of geospatial services are analyzed, it is useful to differentiate between the client software and the server software in order to understand how to improve these systems at each level.

This PhD thesis explores the issues involved in the standards used for constructing spatial data infrastructures and then focus on the portrayal and visualization services

and ways for improving their performance by implementing tile strategies and adopting JPEG2000 compressed formats.

1.B.1.4. Spatial data infrastructures

The precise definition of a Spatial Data Infrastructure (SDI) has been discussed extensively in the literature (Chan *et al.*, 2001). In this paper we use a simple definition that reflects the different senses of the elements of an SDI:

"An SDI is an integrated system that joins together environmental, socioeconomic, institutional databases, etc (horizontal link) and provides an information flow from local or national levels and eventually to the global community (vertical link)" (Coleman *et al.*, 1997)

This definition tells us that SDI are designed to integrate all the stakeholders, to generate geographical data flows in all directions and keep everyone on track. An SDI can be seen as a virtual repository of data that is easily accessed by professionals so they can view, download and work with the data (Nebert, 2004), a market in which companies can obtain the data they need to generate new sources of added value and then sell them in the infrastructures (van Loenen, 2009), or as a place where users can generate new data voluntarily and actively contribute to the growth of the infrastructure (Budhathoki *et al.*, 2008). The words "data" and "information" in this context mean "digital geographic data", although other types of data can also be included. However, and to simplify the explanation, we do not intend to engage in a discussion on the boundaries between data and information, but rather ask the reader to have an open mind and consider the two terms as almost synonymous.

In the first generation of SDI there was the tendency to generate a large container as a data repository for concentrating the available geographic information (Alameh, 2003). It soon became obvious that this approach creates huge maintenance and data updating problems and also requires a large storage capacity. The second generation of SDI focuses on creating a community and a huge active directory of metadata, actors and data catalogues (Figure 3). In this second distributed approach it is paramount to adopt standards for improving interoperability. The European spatial data infrastructure, supported by the INSPIRE directive, follows this second approach (Bernard *et al.*, 2005). Although there are two generations of SDI, the basic goal has not changed significantly over the past ten years (Craglia *et al.*, 2008). This PhD thesis argues that the reason for this is that the real goal of the SDI has not yet been achieved, either because the available solutions are not fully adopted, or because they are being implemented in an incomplete and/or inefficient way. A demonstration of the dissatisfaction with some of this these systems has produced is that some recent initiatives in online mapping have returned to the first generation of SDI methodologies, such as http://geocommons.com.

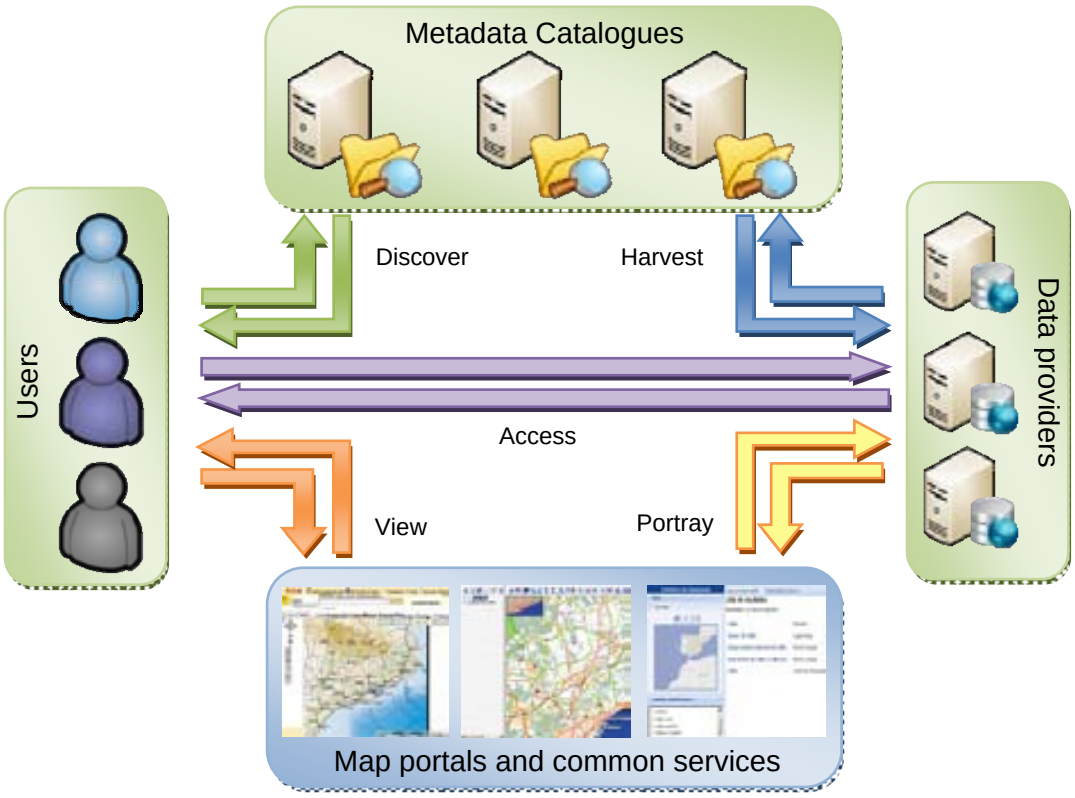


Figure 3: Basic components of the second generation of SDI (source: own preparation).

Chapter 2 of this PhD thesis analyzes the current situation and possible improvements for the SDI phenomena.

The same technology used in the SDI can be used in less structured systems such as the Systems of Systems (SoS). In these systems, various spatial data distribution systems are joined together in a larger system. Therefore, the connection effort to the SoS should be low and based on existing interoperability agreements. The Global Earth Observation System of Systems (GEOSS) is a global system composed of a pre-existing set of Earth observation[24] systems, which makes it possible to detect duplications and encourage the creation of initiatives to fill the gaps in current systems (Butterfield *et al.*, 2008). The system uses a single entry point called GEOPortal, plus a redistribution system with a satellite data communications network called GEONETCast.

1.B.1.5. The Open Geospatial Consortium (OGC)

The OGC is an international organization composed by government agencies, universities, companies and research centres, that aims to promote the usage of open standards and technologies in the context of the geographic information systems and technologies. The OGC acts within three programs: the standards specification program, the interoperability testing program, and the adoption program. In the

[24] The *Earth observation* concept no only refers to *remote sensing* but also includes *in-situ* sensors, etc.

specification program, the standards working groups elaborate documents that standardized independent aspects (such as map services, metadata catalogues, etc.) by consensus, while the domain working groups discuss how to improve interoperability of geographic information from professional sectors or for certain specific topics (such as meteorology, smart cities, etc). Despite the initial interest in the standards for applying programming interfaces (API) in different languages (such as the Simple Features for SQL), which were developed by the OGC over more than a decade, the emerging SDI and their need for interoperable and distributed platforms on the Web is one of the reasons for the success of Web service standards that specify the coding and data standards as far as possible. Web services follow a client-server system that uses a common communications protocol that performs small interactions in which the client requests the remote server to execute an action and waits for the response in the form of a computer file. To make the request, clients send messages encoded in the server's Web address (adding key-value pairs, KVP[25]) (Figure 6) or write small XML documents including the parameters of their request[26].

The OGC has not introduced all of these strategies, but OGC standards describe how the messages should be sent by the clients and how the geospatial services described above will respond. In addition, all services share the OGC GetCapabilities request and a common mechanism for versioning negotiation. The response to these requests depends on the type of service, but it is usually a document encoded in a specific XML dialect. Besides service standards, the OGC has also developed standards for encoding data (to be displayed in virtual globes: KML; for geo-referenced vector data [but not only]: GML; for sensors observations and measurements: O&M; for describing the sensors: SensorML; etc), all within the OGC framework that defines relationships between different standards, such as abstract, for implementations, and profiles. The aim of this is to facilitate making better decisions, for example, for preventing or mitigating the effects of disasters, and planning social development better. Annex I of this PhD thesis contains a detailed description of the OGC, its activities and standards.

To promote a new standard within the OGC it is necessary to have the support of three OGC member organizations, form a working group and begin the process of writing the standard. Once the document has been produced, the draft must be open to public examination for a month, the comments collected and incorporated in the best possible way and then finally the document is submitted for a vote among the OGC principal members. In addition, during this process three reference implementations must be generated. Chapter 4 of this PhD thesis presents the WMTS standard that was written in the WMS standards working group in the interoperability program. It was tested in the OWS-6 in the testing program, in which the three reference implementations were developed that are required for final approval. This was given in early 2010

[25] The key and value pairs are added at the end of the server's address by adding the '?' symbol and then a sequence of the key name, a '=' symbol and its value, another key, a '=' and its value, etc, which are separated by the '&' symbol. An example of this notation can be seen in figure 6.

[26] When the client or the server have special need such as security or encryption, messages can be sent in a Simple Object Access Protocol (SOAP) message that provides a system that combines the body of the message with standard encryption and security mechanisms that are delivered together.

1.B.1.6. Service oriented architectural style

The client-server system uses a protocol to communicate. The service-oriented architectural style[27] (SOA) is used to makes remote procedure calls (RPC)[28] (zur Muehlen *et al.*, 2005). Therefore, we can call a map service to generate a custom map, or we can make a call to a feature server to give us a list of features that meet a certain set of conditions. In principle, most of the OGC service standards follow this style. Conceptually, this architectural style gives greater weight to the action that is remotely executed (continuing with the same examples, a GetMap and a GetFeature requests, respectively) than the resources being manipulated. This has led to the actions being separated into different standards, grouping them by the type of entities they work with (continuing with examples, WMS and WFS respectively). In practice, however, sometimes metadata servers do not connect well to the map services or to download servers making it difficult for the user to move seamlessly from the discovery phase of the evaluation phase to accessing information. Another drawback of this architectural style is that the data is hidden behind the services and it is often not, or partially, accessible (Kim 2004).

1.B.1.7. Resource oriented architectural style

The traditional hyperlinks on the Web have led to a resource oriented architectural style. This style is gaining fans in the geospatial world because it provides a limited common set of operations that can be applied to different resources, which is called a uniform interface. Each resource has a unique identifier, which facilitates sharing and updating it. Moreover, resources are related to each other through links. The resources can not be recovered because they are abstract ideas but the resource representations (which are associated with a computer data format) can.

One type of resource oriented architecture is called the REpresentational State Transfer (REST) (Fielding, 2000). This is the architectural style that the Web uses. There are basically four operations (also called methods, or verbs): (GET = get; POST = create, PUT = update, DELETE = delete) (Mazzetti *et al.*, 2009). What is most interesting about REST is that resources are related to the model in a natural way and receive identifiers that follow a common browsing pattern.

1.B.1.8. The MiraMon GIS

Since the arrival of general-purpose GIS packages in the eighties (Antenucci *et al.*, 1991), many manufacturers have created their own geospatial software. In the mid nineties it became necessary to integrate GIS platforms into the Windows operating system appeared, which made developments independent of the video card and printer. It is in this context that MiraMon began to be developed, in 1994. MiraMon (Pons, 2002) is a Geographic Information System and Remote Sensing software that

[27] The OGC introduced the expression *service oriented architectural style* to avoid conflicting with concepts set up previously that resulted in confusion with commercial brands.

[28] In this text we use *remote procedure call* (RPC) and *service oriented architectural* (SOA) style as synonymous terms, but in reality the SOA style is a more general concept than a RPC.

makes it possible to visualize, query, edit and analyse raster data (remote sensing, orthophotos, digital terrain models, conventional thematic maps with a raster structure, etc.) and vector data (topographic and thematic maps that contain points, lines or polygons). It also connects to OGC services (eg WMS and WMTS) and geographic data sources organized in tabular format, for example in the context of conventional databases, more or less adapted to the needs geographical information. The non-spatial attributes are stored in relational database tables. MiraMon is developed by several members of GRUMETS, a research group formed by members of the Autonomous University of Barcelona (UAB), members of the Centre for Ecological Research and Forestry Applications (CREAF) and members of the Biological Station of Doñana (CSIC). MiraMon is primarily a desktop software that runs on Windows operating systems developed in C language, although there are parts that are intended for servers or clients on the Internet, which are mentioned bellow.

What is considered to be the earliest prototype of an Internet map service, the Xerox PARC Map Viewer, appeared in June 1993. The activities of the OpenGIS Consortium[29] begin in 1997 (Doyle, 1997). In April 2000 it organized its first testbed, called the Web Mapping Testbed, which led to the publication of the first version of the OpenGIS WMS interface. In 2001, MiraMon started a project in a competitive call for advanced communications within the second generation of the Catalan Internet Project. This was the beginning of the MiraMon Map Server (MMS), a software component that provides Web services following protocols and standards set by the OGC, including WMS, WMTS, WFS and WCS. The server application is a CGI[30] executable type, developed in C language, which can be installed directly in a Windows Web services (Internet Information Server, Apache, etc). The MiraMon Map Browser was also developed at the same time that the server. It is a JavaScript thin client that can make requests to WMS and WMTS map services. It also provides data downloading following the WCS protocol and supports WFS and WFS-T (currently limited to point feature types).

The MiraMon development group has 18 years experience in programming, and the program is constantly evolving and being developed further. This is particularly useful for implementing international standards that evolve over time. During the research described in this PhD thesis, numerous implementations of standard clients and servers have been performed. This work would not have been possible without these foundations and previous experience. WMTS implementations (chapter 4) have been carried out in the platforms described above and we participated in the OWS-6 Interoperability Experiment of the OGC to demonstrate the interoperability of three WMTS servers and two WMTS clients. Moreover, the section 1.B.2.1.2 introduces a portal that documents the end user contributions, which is described in more detail at the end of chapter 2. In the section 1.B.1.7 we present implementation of the World Wide Hypermap, which is described at the end of chapter 3. Both implementations have also been carried out with MiraMon technology.

[29] Latter the organization changed its name to the current Open Geospatial Consortium.

[30] CGI is one of the older approaches and transversals to develop server applications. Although Windows internet services support other technologies considered more efficient (such as ASAPI and .NET), resulting developments are too platform depended and our research group avoid their use.

1.B.2. Motivation of the thesis

Spatial data infrastructures are an excellent platform for implementing standards for spatial data. This PhD thesis is motivated by the experience of the GRUMETS group in the implementing and deploying technologies for the distributing data on the Internet, such as the MMZ format (Pons, 2000) and the MiraMon map server and browser. It is based on the experience gained in implementing geospatial standards in the context of spatial data infrastructures (especially for the Catalan SDI, IDEC) and also the work carried out in the past four years within the OGC working groups and, more recently, the GEOSS Standards and Interoperability Forum (SIF). In recent years, we have detected several problems in the standards currently used for SDI development, in general and specifically in mapping services. This PhD thesis aims to verify and locate these problems and provide theoretical discussions on critical performance measurements as well as implement some solutions.

1.B.2.1. Conceptual and architectural issues in implementing of standards for distributed spatial data infrastructures

The section entitled "Use case 1: accessibility to healthcare center" in chapter 2 (Maso *et al.*, 2012), presents a relatively simple use case in which the Catalan Spatial Data Infrastructure (IDEC) was used to locate the resources necessary for studying population accessibility to health centres (Olivet *et al.*, 2008). This use case showed that:

- A part from search and visualization tools, we need other data processing tools based on recognized standards, such as WPS, which have not yet been developed.
- Some services are available as Web portals for users, but there is no interface to use these services from other services in an automated work flow (Maso *et al.*, 2011b).
- It is not possible to return information on the problems with the data search system in order to request corrections or make contributions.
- There was no easy way of integrating the data and results of our study back into the SDI.
- In many cases, if you know the actors involved in the cartographic production of the region, you obtain the best results by contacting them directly instead of using the SDI catalogue. In addition, the catalogue was often missing important actors and map products necessary for making the desired study, even though they do exist.

These observations do not only hold true for the IDEC, as many other SDI have similar problems.

1.B.2.1.1. Current status of the data infrastructures

Considering the current situation, it seems that the conceptual scheme adopted by the SDI is correct, but an in-depth review of all the aspects involved in implementing it is necessary in order to suggest and make improvements in the areas identified as

deficient. This is the main motivation of chapter 2 of this PhD thesis. However, until improvements are applied, the end users themselves could help improve the situation by contributing with their own knowledge and experience about the data, as explained in the following section.

The architectural style adopted by the current SDI and recommended by SDI standard CEN (CEN 15 149) is the service oriented style. In the Section 1.B.1.7 we propose using a resource oriented architectural style, which retains many current aspects but has some additional benefits that, if adopted, could help to correct some of the current problems.

1.B.2.1.2. End user feedback

Due to the institutional character of the spatial data infrastructures, one aspect that has not been given enough attention is the contribution made by end users. Budhathoki *et al.* (2008) and Paudyal *et al.* (2009) both state that the contribution of Volunteered Geographic Information [VGI] and Web 2.0 initiatives could be the main feature of the third generation of SDI.

In fact, the creation of information by volunteers is not a new thing. Volunteers have traditionally helped domains such as ornithology and meteorology to create or enhance the information available in specific disciplines. New technologies have helped to simplify the coordination mechanisms of this information dramatically. Recent examples are the creation of the street and public roads dataset, OpenStreetMap (Haklay *et al.*, 2008) and Weather Observation Website of the British Meteorological Office (Morris, 2012) (Figure 4b).

Currently, the SDI portals mainly focus on providing tools for querying centralized metadata catalogues (that are maintained by the infrastructure support centres) and decentralized data visualization tools (that use the view services owned by the providers). In the second generation of SDI, there is not central data download service. In theory this is because metadata catalogues should provide enough data for finding the necessary information, evaluating this information, choosing from among the data results (including the description of the data model and the information quality), and finally gaining access to the original data producer. However, in practice, in order to make it easier for producers to enter the infrastructure, only a very simple list of items is required to enable data discovery is mandatory[31]. For example, it is not required to have an access system to the information that the metadata describe (either a URL or a data access service) and it is also almost impossible to provide a description of the data model[32]. In addition, SDI rarely incorporate a mechanism for ensuring that the metadata contained in the catalogue really represent all the datasets available from the institutions that are part of the infrastructure. This is often frustrating for the users, who see that the metadata catalogue search results do not match all the available information, the description of the information is not sufficient for determining which product is useful for them, and access paths are missing or

[31] This basic information is known as the *core metadata*, which is composed of 11 metadata elements from among 409 elements defined in the ISO19115.

[32] Indeed, the data model is not described in ISO19115, but rather in ISO 19110, which is almost ignored

incomplete. These problems occur in both the regional infrastructures and global infrastructures such as GEOSS (Diaz *et al.*, 2012).

The possibilities of VGI for enriching SDI have been identified by some authors but there are only a few works that actually contain practical implementations of this approach (Goodchild, 2007). One of the most important is called GeoWiki (Fritz *et al.*, 2009) in which users can carry out quality controls of various land cover and land use maps by intercomparison (Figure 4a). Following similar strategies, users could contribute to the SDI metadata catalogues, enriching them, acting as data quality control, and also providing new information or expressing their ratings and opinions. To make this possible, we need new tools for creating content and facilitating user interaction with the system, enabling the user to make the aforementioned contributions, which can then be added to the SDI portals or to other related sites. The section "Use case 2: Web 2.0 metadata user comments: IDECTalk" in chapter 2 proposes an implementation of user feedback.

(a) (b)

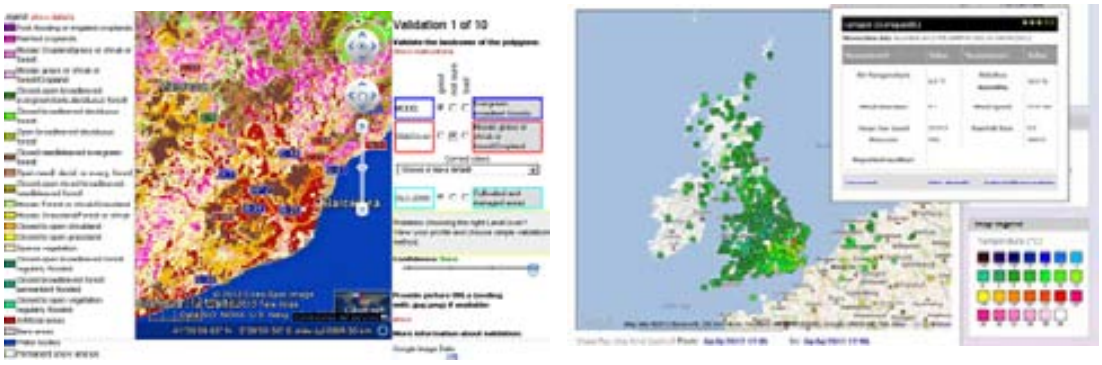


Figure 4: VGI portal examples: (a): GeoWiki (font: http://www.geo-wiki.org/) (b): MetOfficeWOW (font: http://wow.metoffice.gov.uk/).

1.B.2.1.3. The hypermap and its four limitations

In 1990 by Laurini *et al.* (1990) established the hypermap concept by extending the hypertext to digital maps. Hypermaps are geo-referenced multimedia systems that can relate individual components that that are linked to each other and to the map (Kraak *et al.*, 1997), and thus features on a map can be linked to information of any kind (text documents, pictures, sounds, videos and other multimedia content) through hyperlinks. These links are represented on the screen by buttons appearing as icons/pictograms or typographical enrichments in the text, etc. (Boursier *et al.*, 1992).

The largest limitation of the hypermap is that all queries must be previously defined and “prepared”, i.e., links must be established between entities and users can only follow the particular paths laid out previously (Boursier *et al.*, 1992). In addition, resources are linked to other resources by pairing local identifiers, and thus the scope of their relationships are limited (Yuan *et al.*, 2000). Another criticism is that the underlying data model is extremely poor because it only includes the concept of hyperlinks among entities (Boursier *et al.*, 1992, Voisard 1998) and has no semantics that identifies the purpose of each hyperlink. Moreover, hypermaps are limited to

retrieving resources (Voisard, 1998) and are not able to create, update or delete resources.

The hypermap concept can be extended following the REST pattern applying it to geographical resources, so that we can overcome the limitations of the hypermap outlined in the previous chapter. Thus, in a possible REST[33] implementation, a resource is a geographic feature. Another resource is a geographical feature collection, which may have different representations in different formats (one of them could be a GML file, another a SHP file, etc). A feature collection is related to its metadata, and can lead to new types of resources in the form of a map. Chapter 3 of this PhD thesis discusses how we can apply the resource oriented architectural style based on REST in a distributed GIS, so that geographical resources are managed better.

1.B.2.2. Specific issues of Internet map browsers and uses cases

In the previous sections we presented the concepts that we will use in order to discuss more general issues in the standards, this chapter focuses on one particular standard type from those introduced in the section 1.B.1.3: portrayal and visualization services. These services are, without a doubt, the most commonly used services in digital cartography because they show the information immediately in a visual way for all users and hide the complexities in the data structure. The impact of these services is so high that in many cases they have overshadowed the need to distribute data in the original format, as noted in chapter 2.

The standards for portrayal and visualization (Figure 5) are:

- OGC Web Map Service (WMS) approved in 2000 (de la Beaujardiere, 2004), which was later adopted as ISO 19128.
- OGC Web Map Tile Service (WMTS) (Chapter 4), approved in 2010.
- The ISO 19117 Geographic Information - Portrayal, which defines the basic concepts about symbolisation.
- The OGC Symbology Encoding (SE-OGC) (Muller, 2006).
- The OGC Styled Layer Descriptor (SLD) (Muller *et al.*, 2005), which allows applying symbolization to be applied in a WMS service.
- The OGC Geography Markup Language (GML) version 3 can also encode a default symbolization.

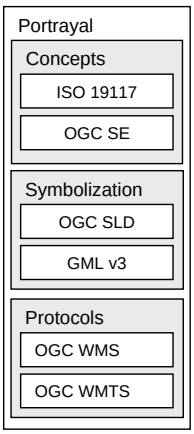


Figure 5: Visualization standards (source: Figure 2, chapter 2).

Standards for portrayal and visualization were originally designed for creating maps (cartographic representations of the image resolution required for displaying on the

[33] Implementations that follow the REST architectural style are often called RESTful implementations.

output device) for viewing portals embedded in Web pages. However, they are now applied to other software, from desktop GIS to mobile device applications.

The WMS standard defines a mandatory operation (GetMap) for requesting and getting a map of an area defined by its extent and size in pixels, from the data of one or more layers (Figure 6). Generally, this representation is encoded in a common format in Web browsers (image/jpeg or image/png). It also provides an optional operation (GetFeatureInfo) for obtaining more information about a point on the map, usually in text format, which is typically used to implement queries by location (Bermúdez *et al.*, 2012).

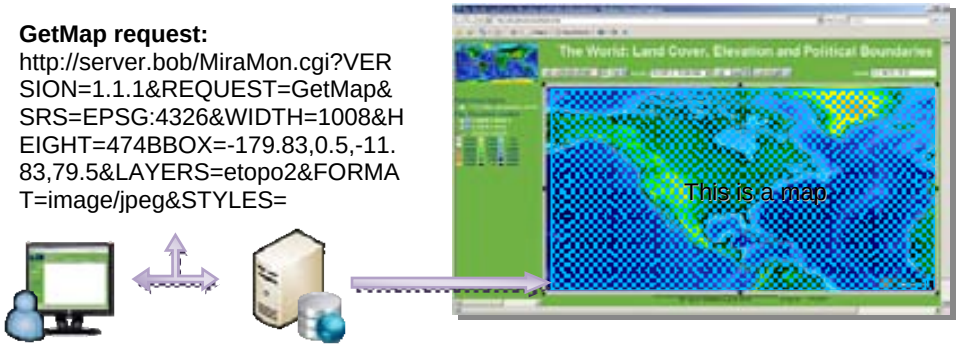


Figure 6: WMS clients commonly use GetMap operations to request the whole view-port from the server (source: own preparation).

1.B.2.2.1. Rigorous analysis of map server performance

A major problem in many implementations of WMS map servers is their slow response, especially if a scale that is very different from the original scale of the data is requested, or if there are many concurrent requests (Yang *et al.*, 2005).

To evaluate different products rigorously, manufacturers need to adopt comparable conditions and strategies. These strategies focus on improving the performance of the implementations and determining the causes of performance degradation in specific implementations for concurrent requests.

One of the causes of performance degradation in the WMS is more conceptual than technical and lies in the flexibility of the standard itself, which hardly encodes two requests that are exactly the same. This prevents old requests and results being recycled, and also prevents users' needs from being anticipated and prerendered views being saved.

In this regard, chapter 6 discusses objectively the performance of different products from different manufacturers that serve maps using the WMS protocol, and some variants that serve tiles, in relation to the number of concurrent requests and the scale requested.

1.B.2.2.2. Maps cut into tiles

An alternative for improving the performance of map servers is to define a discrete set of zoom levels (scales) at which requests will be visible and define a tile pattern for each zoom level (Quinn *et al.*, 2010). The server can only respond to requests for a limited set of small maps called tiles (Figure 7). This approach ensures that different caching mechanisms that may exist in both the server and the client, and also at an intermediate point of the network, can work more efficiently because there is a greater likelihood that some tiles will be requested several times. Furthermore, in these circumstances it is also possible that the server has fully prepared all possible tiles.

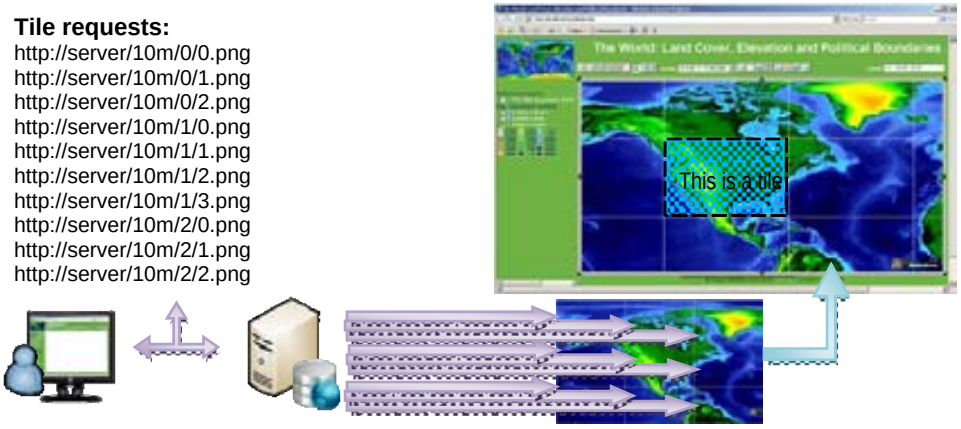


Figure 7: A tile client makes simultaneous requests for all tiles that cover the view-port, except when they are already in the client *cache*. It also hides the tile areas that are not part of the view-port (source: own preparation).

This approach is used by products or services such as TileCache (TMS), Google Maps and TerraService (Barclay *et al.*, 2006), and also the WMS-C OSGeo standard. Each system defines its own set of zoom levels (Figure 8b), the tile pattern from each of the zoom levels (Figure 8a), its own way of exposing this pattern and its own way of recovering a tile. The existence of different alternatives is an interoperability issue, because only a few clients can read all the existing variants.

It became necessary to develop an international standard validated by leading software manufacturers and major government agencies. Therefore, we created the WMTS standard contained as Chapter 4 in this PhD thesis. The creation of an OGC standard follows a consensus-based process, as explained in the section 1.B.1.5. The document was approved with 40 yes votes from the 81 voting organizations with only 1 no vote.

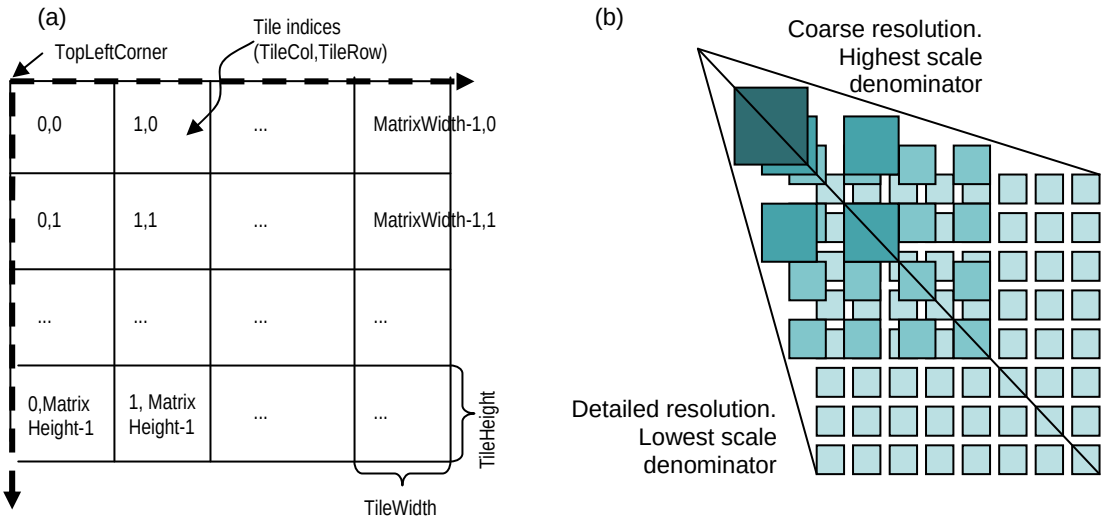


Figura 8: (a) Tile pattern definition (source: Figure 2, chapter 4) and (b) zoom levels for a map tile system, in this case the WMTS OGC standard (source: Figure 3, chapter 4).

1.B.2.2.3. The JPEG2000 format

Another alternative for improving the performance of WMS servers is to optimize the internal storage format and the data extraction algorithm to minimize the response time for each map request.

The traditional JPEG format is one of the most used formats in WMS and WMTS map servers because it can be shown directly by current Web browsers. The JPEG2000 format has many advantages over traditional JPEG (Zabala, 2010), which also affects its speed because it provides direct access (random) to parts of the image. This international standard (ISO15444-1) also supports lossless and lossy compression, which is a detail that may be important in contexts where the best quality in vital. JPEG2000 offers better quality at the same compression ratio (for example, there are no 8x8 block artefacts like the ones in classical JPEG) and is designed for easy screen viewing of large images (particularly those with continuous tones). Therefore, images at different resolutions and sizes from the same compressed file can be recovered (traditional JPEG images can only be recovered at a fixed resolution and by uncompressing the whole file). However, to get all these benefits, the computational cost of the algorithm is much higher, so that the complete compression time (and, in less degree, the decompression time) for JPEG2000 image is significantly larger than for classical JPEG images. For all these reasons this format is appropriate for accessing small areas in very large images.

To simplify it greatly, the wavelet transform of the original image cut in rectangular codeblocks[34] is stored within a JPEG2000 image (Figure 9). The wavelet transform is reversible so it is possible to obtain the original image at the original resolution but

[34] It is easy to get confused between a *codeblock* and a tile: A *codeblock* is a rectangular fragment of the transformed *wavelet* space but a tile is a rectangular fragment in the original non-transformed space.

also extract a lower level of detail more easily by decompressing only a fraction of the codeblocks.

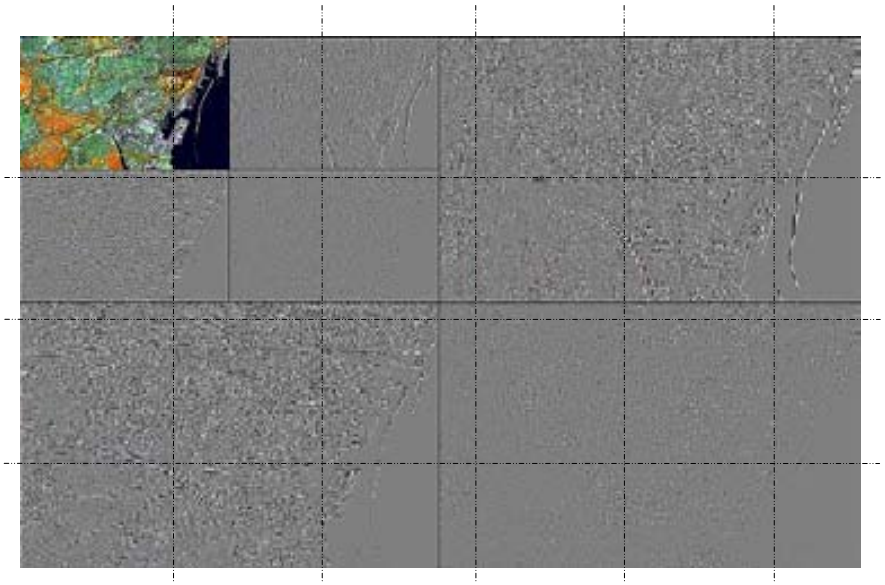


Figure 9: *Wavelet* transform of a false colour composition of a fragment of a Landsat image of the Barcelona metropolitan area. Solid line, three transform resolution levels. Dashed line: *codeblocks* division (only the first two *codeblocks* are required to decompress the whole image to ¼ of the original de resolution) (source: own preparation using fwt2d software[35]).

In addition, before it is transformed and compressed, the original image can also be cut into tiles in the non-transformed space, which are then compressed independently but can be stored in a single JPEG2000 file (Figure 10). Furthermore, the ISO15444-2 standard defines a way of including metadata within JPEG2000 image files as an XML box that is used by the OGC GMLJP2. Moreover, the ISO15444-9 standard defines a communications protocol for transmitting incremental information encoded in JPEG2000 format. Given the similarities between the tile systems and the JPEG2000 format, it makes sense to consider whether the JPEG2000 format could an alternative or complement to WMTS.

Chapter 5 explains the different alternatives for combining JPEG2000 with different OGC standards and proposes that the JPEG2000 format is a good format for storing the original data in the WMTS servers if compression is carried out in a certain way.

[35] Windows software coded by Chesnokov Yuriy. 20 October 2007. http://www.codeproject.com/Articles/20869/2D-Fast-Wavelet-Transform-Library-for-Image-Proces

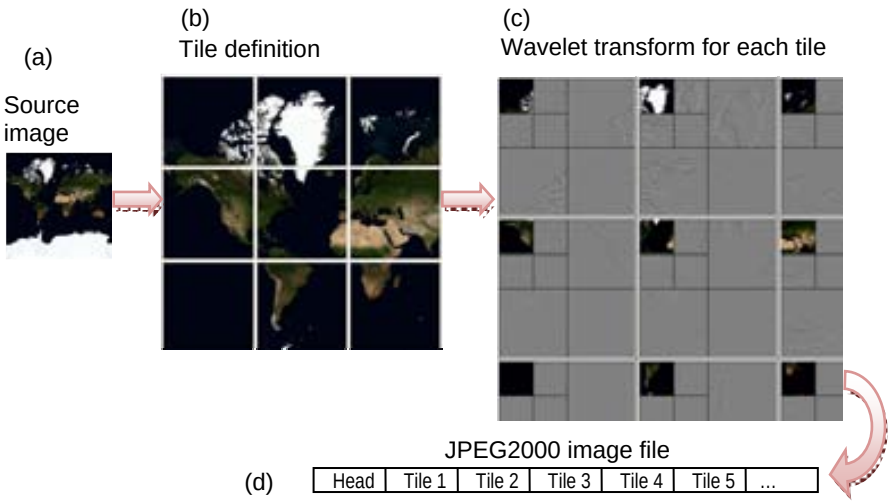


Figure 10: The original image (a) can be cut into tiles (b) that are independently *wavelet* transformed (c), cut into *codeblocks* and incorporated in the byte sequence in the JPEG2000 file (d) (Figure based on: Figure 7, in chapter 5).

1.B.3. Thesis objectives

The overall objective of this PhD thesis is to study the current status of Internet data models for geographic information (including remote sensing data) and the implications and difficulties involved in their practical implementation and deployment, especially in the context of SDI as well as propose corrections for some of the problems found.

This general objective can be divided in the following specific objectives:

The first specific objective is to review the standards used in the SDI, the implementation process and the difficulties observed in the technologies that use these standards. We propose specific improvements at both the service and the client levels in order to help to evolve the status of the SDI.

The second specific objective is to define a new technological framework that restructures the current architectural style used in the SDI, with is currently a service oriented, into a different resource oriented architectural style in order make better use of the Internet, relate resources better and incorporate other collectives, such as for example standards used in open data policies, volunteered geographic information and mass market map browsers and virtual globes.

The third specific objective is to contribute to improving the performance of implementations of standards-based map viewers. We propose strategies for incorporating the JPEG2000 format and a new tiled map service called WMTS.

The fourth specific objective is to incorporate new standards into the MiraMon software technologies (the desktop version and also the map browser and map server)

(Pons, 2002) in order to validate them by using them. This process will allow systems to be designed that optimize the algorithms that implement the above proposals.

Finally, the fifth specific objective is to analyze quantitatively the different map services products (including developments incorporated into MiraMon) and assess to what extent the proposed improvements contribute to increasing the performance of the implementations.

1.B.4. Document organization

The previous introduction provides the reader with the general background needed to understand the different chapters of this PhD thesis. It is also available in Catalan in the subsection 1.A. It is necessary to note that this is a PhD thesis in the form of a summary of publications, so that each chapter contains its own abstract and its own individual introduction, and therefore the reader can obtain extra introductions to the specific aspects covered in this chapter.

The PhD thesis is divided into two blocks. The first consists in general aspects that describe various problems involved in applying standards to SDI and also makes specific proposals for fixing these problems. It also suggests a new REST architecture for creating a World Wide Hypermap. The second block presents specific aspects for improving map services. In this block, we propose adding JPEG2000 or including strategies based on tiles and some implementations performance measurements.

This document is presented as a summary of publications and contains five publications found in the central chapters that describe in detail the aspects introduced in this section, as well as a chapter with the summary of the results and the conclusions in order to provide an overall view of all the achievements. An additional publication is included in Annex I as it is currently other final review process and therefore could not be included in the main body of the PhD thesis; however, we believe that it is of interest in the context of the research and complements the set appropriately

The presented aspects are discussed in a little more detail in the following overview.

Some general aspects of the geospatial standards detailed in this PhD thesis are:

As an introductory complement Annex I of this PhD thesis presents an overview of the main OGC standards and their foundations, providing a more practical approach beyond the pure repetition of the standards. It also includes original figures. We believe it represents a good introduction to more specific topics developed in the PhD thesis. The publication has been sent to be a chapter in the book "Fundamentos de las IDE" which will be published by UPMPress and is edited by Miguel A. Bernabé-Poveda and Carlos M. López-Vázquez (Bermúdez *et al.*, 2012)

Chapter 2 of this PhD thesis reviews the current generation of SDI and proposes ways of improving performance. It was published in the *International Journal of Geographical Information Science*, indexed by JCR-SCI (Masó *et al.*, 2012a). It also presents two use cases in more detail: one that uses the Catalan SDI (IDEC) for a

studying the accessibility to health centres, and one that look at an pilot system in which users can make comments to the IDEC catalogue records.

Chapter 3 of this PhD thesis suggests extending the hypermap concept to the World Wide Hypermap (WWH) by applying the new resource oriented architectural style and the REST architecture that solves some problems discussed in chapter 2. It also shows how to implement the architectural style in the MiraMon software. It was published in the *International Journal of Digital Earth*, indexed by JCR-SCI (Masó *et al.*, 2012b). This chapter is supplemented by Annex II which provides a list of resources and operations defined in the WWH concept.

The following chapters deal with the particular aspects of map servers:

Chapter 4 of this PhD thesis is an international standard that describes a protocol to expose the tile structure of geographic information in map form (as a portrayal), and requesting and receiving a tile. This proposal was accepted by the OGC in 2010 (Masó *et al.*, 2010a). Its impact can be measured by the 16 citations that have been made in several scientific papers so far (February 2012).

Chapter 5 of this PhD thesis examines how we can improve the performance of map servers by applying the JPEG2000 format. It was published in the *Italian Journal of Remote Sensing* (*Rivista Italiana di Telerilevamento*), indexed by JCR-SCI (Masó *et al.*, 2010b).

Chapter 6 of this PhD thesis analyzes the impact of concurrent requests in the map server implementations from different manufacturers in the sector with either WMS or WMTS. This publication is part of the proceedings of the 2011 international conference organized by the International Academy, Research and Industry Association (IARIA) (Masó *et al.*, 2011).

Chapter 7 is the summary of the results shown in chapters 2 to 6. Finally, chapter 8 lists the conclusions of this PhD thesis. Annex III contains a list of acronyms.

2. TUNING THE SECOND GENERATION SDI: THEORETICAL ASPECTS AND REAL USE CASES

Aquest capítol és una reproducció de: Masó J., Pons X. and Zabala A. (2012) Tuning the second generation SDI: Theoretical aspects and real use cases. *International Journal of Geographical Information Science*. DOI:10.1080/13658816.2011.620570 (en aquest document es referencia com Masó *et al.*, 2012a)

International Journal of Geographical Information Science
i*First*, 2011, 1–32

Taylor & Francis
Taylor & Francis Group

Tuning the second-generation SDI: theoretical aspects and real use cases

Joan Masó[a]*, Xavier Pons[b] and Alaitz Zabala[b]

[a]*Center for Ecological Research and Forestry Applications (CREAF), Universitat Autònoma de Barcelona (UAB), Bellaterra, Barcelona, Spain;* [b]*Geography Department, Universitat Autònoma de Barcelona (UAB), Bellaterra, Barcelona, Spain*

(*Received 22 January 2011; final version received 26 July 2011*)

Spatial data infrastructure (SDI) actors have great expectations for the second-generation SDI currently under development. However, SDIs have many implementation problems at different levels that are delaying the development of the SDI framework. The aims of this article are to identify these difficulties, in the literature and based on our own experience, in order to determine how mature and useful the current SDI phenomena are. We can then determine whether a general reconceptualization is necessary or rather a set of technical improvements and good practices needs to be developed before the second-generation SDI is completed. This study is based on the following aspects: metadata about data and services, data models, data download, data and processing services, data portrayal and symbolization, and mass market aspects. This work aims to find an equilibrium between user-focused geoportals and web service interconnection (the user side vs. the server side). These deep reflections are motivated by a use case in the healthcare area in which we employed the Catalan regional SDI. The use case shows that even one of the best regional SDI implementations can fail to provide the required information and processes even when the required data exist. Several previous studies recognize the value of applying Web 2.0 and user participation approaches but few of these studies provide a real implementation. Another objective of this work is to show that it is easy to complement the classical, international standard-based SDI with a participative Web 2.0 approach. To do so, we present a mash-up portal built on top of the Catalan SDI catalogues.

Keywords: SDI; metadata; web service; standards; data access; data portrayal

Introduction

The precise definition of a spatial data infrastructure (SDI) has been discussed in several papers (Chan *et al.* 2001, Nebert 2004, Wytzisk and Sliwinski 2004, Vandenbroucke *et al.* 2009). For the purposes of this text, we chose to use a simple definition that reflects the different directions of the elements in an SDI:

> An SDI is an integrated system that joins together environmental, socioeconomic, institutional databases, etc (horizontal link) and provides an information flow from local or national levels and eventually to the global community (vertical link). (Coleman and McLaulghlin 1997)

*Corresponding author. Email: joan.maso@uab.cat

ISSN 1365-8816 print/ISSN 1362-3087 online
© 2011 Taylor & Francis
http://dx.doi.org/10.1080/13658816.2011.620570
http://www.tandfonline.com

This definition tells us that SDI implementations attempt to integrate every possible actor to generate data flows in every possible direction and keep everyone in the loop. An SDI can be seen as a virtual repository of easily accessible data that professionals can see, download and work with (Nebert 2004); as a marketplace where companies can obtain the data they need to generate new added value resources that are later sold on the infrastructure (van Loenen 2009); or as a place where users can generate new data collectively and voluntarily and actively contribute to the growth of the infrastructure (Budhathoki *et al.* 2008). The words 'data' and 'information' mean here 'digital geospatial data' even though other kinds of data can also be included (we will not discuss here the differences between data and information). Sometimes SDIs are also called geospatial data infrastructures (Groot and McLaughlin 2000) or geographic information infrastructures (van Loenen 2009).

Geographical information systems (GIS) are one of the main consumers of geospatial digital data. A GIS is a system essentially composed of hardware (currently this also includes communication networks), software, experts and data/information. Moreover, data collection is critical for the success of the system. By making it possible for data to be shared, SDIs provide the fuel for the spatial analytical tools available in most GIS software (Yang *et al.* 2006). With good spatial modelling techniques and the right data, new information will emerge, which will ultimately assist in decision-support tasks and help decision-makers to come to appropriate decisions (Vanderhaegen and Muro 2005). Eventually, this will convince decision-makers to increase funding for SDI development. Nevertheless, the aim of making data accessible to as many people as possible has not yet been fully achieved and the challenge remains of how to develop SDIs that can provide an enabling platform in a transparent manner that serves not only professionals but also the majority of society (Masser *et al.* 2008).

The SDI phenomenon is more than 15 years old and several papers have analysed its different aspects, such as suitability for disaster management studies (Mansourian *et al.* 2006), environmental impact assessment (Vanderhaegen and Muro 2005), marine studies (Wright *et al.* 2003), local planning (Nedovic-Budic *et al.* 2004), geoportals (Beaumont *et al.* 2005), distributed processing (Yang and Raskin 2009), cultural implications (Rajabifard *et al.* 2002b), SDI hierarchy levels (Rajabifard *et al.* 2006) and performance indicators (Scholten *et al.* 2006). There are high expectations for SDIs, but just 5 years ago there were still very few scientific papers on the subject (Bregt and Crompvoets 2004); however, the number of papers is currently growing. Some works have identified the need for new ideas and developments for improving the capabilities of the current-generation SDI resources (Wright *et al.* 2003, Tu *et al.* 2004, Yang *et al.* 2005, Craglia *et al.* 2008), but more research is necessary (Bernard *et al.* 2005a). For example, only a few papers that analyse the current state of SDI development actually propose new improvements that can affect performance and usability. However, new technological tools, like virtual globes, are changing the way people interact with geospatial data (Butler 2006, Craglia *et al.* 2008), as they provide platforms that are more attractive to the general public and mass market than traditional SDI geoportals, although their functionality is limited. The emerging GeoWeb 2.0 phenomenon (Masser 2009) is also opening new possibilities, as it allows user-generated contents (Budhathoki *et al.* 2008) to complement the traditional National Map Agency (NMA) products. Many of these initiatives mobilize user groups that generate continuously evolving, large georeferenced collections of measurements (Goodchild 2007a). Finally, some studies suggest that more mature national

SDI initiatives are not growing as fast as would be expected and often fail to involve local administrations and the private sector, which suggests that the overall health of some SDIs is not in good shape (Crompvoets *et al.* 2004).

SDI generations

Rajabifard *et al.* (2002a) present two approaches to the SDI concept: product and process oriented. The first approach to be defined and applied was the product-oriented approach. This approach concentrates efforts on developing a common and fundamental database that collects all available datasets in a single place. It provides the data with simple ftp or http downloading processes or now through emerging web service architectures (Alameh 2003). The second approach, the process generation approach, focuses more on creating a community and acting as a huge active directory that links metadata, data and people. Modern SDI implementations, such as the emerging European SDI (Bernard *et al.* 2005b) and its member state SDIs, use the second model. Surprisingly, direct data access is losing importance and attention has become focused on metadata and portrayal services. Since this is an evolution of the first SDIs, this second approach is also called 'second generation' in the literature (Masser 1999, Crompvoets *et al.* 2004). This distinction between 'product/data' and 'process' orientation was first established by Keller (1999) in relation to the interoperability concept. In the 'process-oriented' name, the word 'process' does not mean analytical data processing tools or geoprocessing services, but rather it refers to a more generic change so that it is not only the data that are provided (i.e. file downloading systems), but web services that help discover, exchange and harmonize data among SDI actors. Of course, downloading data/information is still necessary, or we would end up with systems providing data that are 'almost useless, poor, or empty of contents' from a real data point of view. Other authors state that a decade of experience of first-generation SDIs and the emerging second generation have enabled us to evaluate the successful factors and determine possible improvements in the current second-generation SDI phenomenon (Kok and Loenen 2005, Maguire and Longley 2005).

There is no consensus on what the third-generation SDI will be. Rajabifard *et al.* (2006) and Masser (2009) propose that sub-national SDIs will play the most important role in third-generation SDIs, and that they will create more opportunities for the private sector; however, Budhathoki *et al.* (2008) and Paudyal *et al.* (2009) believe that third-generation SDIs will be characterized by volunteered geographic information (VGI) and Web 2.0 initiatives. Despite the two generations of SDIs and the third generation to come, the underlying basic aim of the SDI architecture has not changed significantly over the past 10 years (Craglia *et al.* 2008). This article argues that the reason is that the real aim of SDIs has not yet been achieved because the available solutions are not fully adopted or are poorly implemented.

User-side versus service-side improvements in the current SDI generation

The process-oriented approach used in the second-generation SDI can be divided into two parallel sides: a user and a service side. This division is deeply rooted in the Internet client–server architecture but is rarely used in the literature when the SDI state is analysed and improvements are suggested. The user side is closer to people's needs as it provides web pages that allow users to browse through metadata and map representations

(Wytzisk and Sliwinski 2004). The service side provides other services for accessing data directly and for executing and linking processes that can enrich the SDI possibilities (Crompvoets *et al.* 2004). The two sides are complementary, and sometimes improvements and recommendations are needed on the user side, on the service side or on both sides. In the second-generation SDIs, both the user and server sides contribute to building a new infrastructure capable of generating knowledge.

In the past 10 years, standard organizations made many efforts to specify server-side protocols and formats and meanwhile SDI websites and geoportals focused on increasing the potential of the user side and its friendly use (Wright *et al.* 2003, Tait 2005, Yang *et al.* 2006, Goodchild *et al.* 2007, Yang *et al.* 2007). However, this has led the community to partially forget the needs of the service side (Maguire and Longley 2005). In many cases, user-side SDI geoportals are powered by standard web service components and perhaps other proprietary services without providing much documentation on how other services or clients can use them directly. Sometimes, there is a standard web server, but it is presented using a middleware (Middleware tier) that hides the standard entry point from a general user (Nebert 2004). In some other cases, standard servers are inaccessible for security (e.g. health sensible data) or technical reasons (Thompson *et al.* 2009). In other cases, access is open but the URL entry point is difficult to find. Web map service (WMS) servers are perhaps the exception to this, but even in this case some service instances are difficult to find.

Many aspects could be improved in the current SDI implementations; however, some solutions may seem contradictory if we do not take into account that these two sides coexist and need to be improved with different, but sometimes overlapping, technological solutions.

Objectives and structure of the article

The purpose of this article is to analyse if the foundations of SDIs need immediate and complete reconceptualization or rather it is only necessary to reinforce or refocus some aspects of the current-generation SDIs to overcome current problems.

This article reviews the common SDI geoportal components, and then discusses some crucial problems observed in the current-generation SDIs. We propose parallel improvements to SDI technologies and implementations that would allow both users and services to access SDI data, to process them and to generate new products that feedback into the SDI catalogues. Some of these improvements are based on previous literature and others come from our own experience. All the solutions and improvements are classified by topics and summarized in a table. This article also includes two use cases: an example that uses an SDI to try to find and obtain GIS data then incorporate the results of the geospatial study back into the SDI, and a Web 2.0 participatory implementation aimed to complement catalogue metadata records.

Review of SDI geoportal components

From a technological perspective, the main element of an SDI is the geoportal (Bernard *et al.* 2005b), sometimes called a clearinghouse (Crompvoets *et al.* 2004). A geoportal can be defined as an electronic facility for searching for, viewing, transferring, ordering, advertising and disseminating spatial data from numerous sources via the Internet

(Crompvoets and Bregt 2003). The portal should offer at least the following functionalities (Nebert 2004):

- Geospatial data and service catalogue, which makes it possible to discover data and services and collect metadata from the producers.
- Geospatial data visualization, which allows the data to be viewed online. It is based on portrayal services or map services.
- Geospatial data access and delivery, which provides open access to raster (coverage services; Whiteside and Evans 2008) or vector (feature service; Vretanos 2005) data.

These essential services can be complemented by the following functionalities (Bernard *et al.* 2005b):

- Gazetteer, which offers geocoding functionalities for relating a geographic name, spatial code and so on to geospatial coordinates or feature representations.
- Thesaurus and data models, which are specialized vocabularies that will increase semantic interoperability in the future.
- Coordinate transformations, which transform coordinates among coordinate reference systems and help to harmonize data based on different coordinate reference systems.
- Others: classification, statistical analysis, generalization and so on; sometimes grouped under the generic name of processing services.

In order to guarantee interoperability between functionalities and services among the different SDI levels, standards must be carefully considered and followed. The most important geoinformation-related standards are those developed by the Technical Committee 211 of the International Organization for Standardization (ISO/TC211, http://www.isotc211.org) and those developed by the Open Geospatial Consortium (OGC, http://www.opengeospatial.org). OGC has an active interoperability program that tests future standard ideas and develops new ones. Therefore, standards are controlled by a test process that partially guarantees the feasibility of deploying these standards. OGC and ISO/TC211 have an agreement for the development of a series of Industry Implementation Standards based on ISO 19100 and other related standards. Examples of this agreement are the ISO 19115 Metadata standard that was approved in 2003 (ISO 19115) and later adopted by OGC as topics 9 and 11 of the OGC Abstract Specifications, and also version 1.2 of the OGC Web Map Service Standard (WMS) approved in 2000 (de la Beaujardiere 2004) that was later adopted by ISO as ISO 19128.

The list of ISO and OGC standards is a long one and some documents are still under elaboration. For instance, 'ISO 19139: Metadata XML encoding' was released in 2007 (ISO 19139) but the XML encoding for 'ISO 19110: Feature catalogue' is still under development in the Eden initiative (http://eden.ign.fr/xsd/isotc211). Therefore, some client or server implementations of recent standards are not fully compliant or, in the worst cases, there is only test pilot software. Other standards are very popular and there are hundreds of implementations, like OGC-WMS clients and servers (Kolodziej 2003) or ISO 19115 Metadata editors (Rajabifard *et al.* 2009).

Table 1 summarizes the standards that are key components of the SDI geoportals, while Figure 1 shows an architecture example in which these standards are used to connect the SDI geoportal with SDI providers.

Table 1. Most relevant standards for the SDI geoportals.

Geospatial metadata catalogue	**ISO 19115:2003 Geographic Information – Metadata** Defines models and concepts needed to describe geographic data. It defines title, abstract, keywords, responsible parties, reference system, lineage, quality parameters, spatial resolution, geospatial and temporal extend, etc. **ISO 19115-2:2009 Geographic Information – Metadata – Part 2: Extensions for imagery and gridded data** Extension that covers the measuring equipment and production process used to acquire the raw data and the derivation of geographic information from raw data. **ISO/TS 19139:2007 Geographic Information – Metadata – XML schema implementation** Defines how to encode metadata for a dataset in an XML document that can be exchanged or stored in a catalogue. **ISO 19119:2005 Geographic information – Services** Defines models and concepts needed to describe the characteristics of geospatial services, allowing service discovery and chaining. **OGC OWS Common** Defines how to describe the service (title, provider), the service access (URL endpoints and protocols) and the service data or processes offered. **OGC Catalogue service web** Defines a syntax and protocol to get and filter metadata about data or services.
Geospatial data visualization	**OGC Web Map Service (WMS)** Defines a syntax and protocol to get maps that are representations of a fragment of dataset in commonly used image formats at screen resolution. **OGC Symbology Encoding (SE)** Defines a language and XML encoding to describe rich renderization strategies to portray a dataset in a map. **OGC Style Layer Descriptor (SLD)** Extension of WMS that describes a syntax to dynamically send a user-defined symbolization (and XML SE file) to a layer in a map server and apply it.
Geospatial data access and delivery	**OGC Web Feature Service (WFS)** Defines a syntax and protocol to get parts of a vector dataset as GML files. **OGC Geospatial Markup Language (GML)** Defines a language and XML encoding to describe spatial features and properties. **OGC Web Coverage Service (WCS)** Defines a syntax and protocol to get parts of a raster dataset in a GIS raster format. **OGC Sensor Observation Service (SOS)** Defines a syntax and protocol to get a description of a sensor, the variables it measures and the recorded values.
Gazetteer	**OGC Gazetteer Service. WFS Application Profile** Defines how to use a WFS service to implement gazetteer functionality.

(*Continued*)

Table 1. (Continued)

Thesaurus and data models	**OGC Geospatial Markup Language (GML)** To encode features, the GML language requires a GML application schema that defines feature types. It also defines rules for encoding dictionaries in XML. **ISO 19110:2005 Geographic Information – Methodology for feature cataloguing** Defines models and concepts needed to describe feature types, a flat list of attribute types and also operations and associations over feature types.
Coordinate transformations	**OGC Coordinate Transformation Service (CT)** Defines a syntax and protocol to transform data from one coordinate reference system to another.
Other	**OGC Web Processing Service (WPS)** Defines a syntax and protocol to describe and execute geospatial processes.

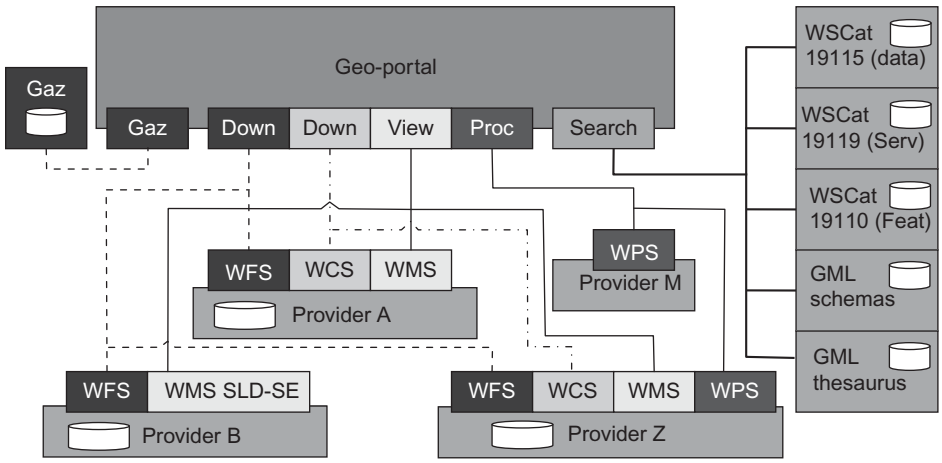


Figure 1. Geoportal architecture of distributed web services showing the gazetteer (Gazz), download (Down), viewing (View), processing (Proc) and search (Search) portal components with connections to some service providers. Adapted from a European geoportal figure (Bernard *et al.* 2005b).

Improving the user and service sides in current SDI implementations

As mentioned above, this article discusses recommendations for improving SDI implementations, such as adding more functionalities, enhancing interoperability or improving performance. These recommendations have been classified into the following topics: *metadata about data and services*, *data models*, *data download*, *data and process web services*, *portrayal and symbolization* and *mass market*. We consider the user side and the service side in parallel. The relevant standards and protocols that are mentioned in this discussion can be seen in Figure 2. The following paragraphs discuss many recommendations, but Table 2 contains a more complete set.

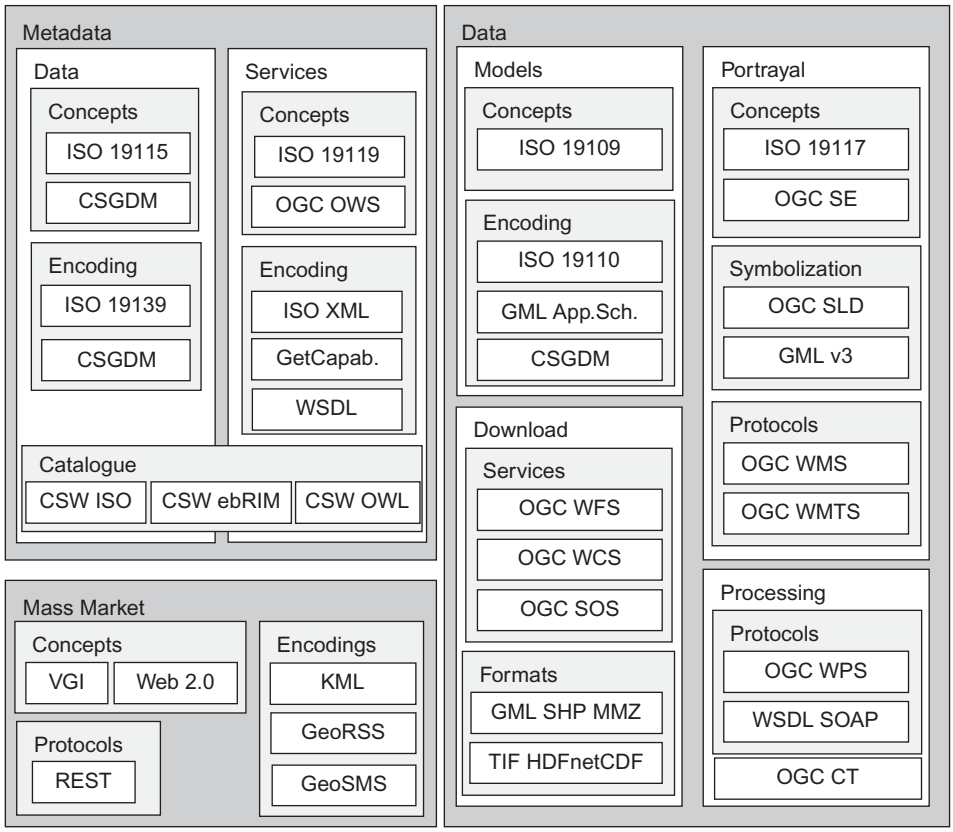


Figure 2. Standards and protocols that are considered in the discussion on improving SDI.

Improving metadata about data

Metadata can refer to data and services, among others. This essential element for data discovery is generally stored in catalogues for efficient search purposes. Metadata standards for data have been used for more than a decade (starting with the Federal Geographic Data Committee metadata standard – Content Standard for Digital Geospatial Metadata (CSDGM); Metadata Ad Hoc Working Group 1998), but metadata catalogue standards are more recent and they come in a variety of versions and profiles of both CSW ISO 19115–19139 and CSW ebRIM. This generates some confusion (Nebert *et al.* 2007) and makes it difficult to set up decentralized catalogues that actually work. Some of these interoperability problems can be overcome by a middleware software that acts as a protocol translator or broker, such as the Catalogue Connector, which translates a generic user request into each particular catalogue dialect from a single portal interface (Pascual *et al.* 2009).

Nowadays, almost all current SDIs carry out the discovery search in their centralized catalogue, which compiles all the available metadata records (Bernard *et al.* 2005b). This is a good solution for making searches but it complicates updating. Indeed, when a data producer comes with a new dataset or a simple metadata update, they have to communicate it to several SDI facilities or wait for catalogues to reharvest. The hierarchical organization of the national, sub-national and local SDIs (Rajabifard *et al.* 2002b) can simplify

Table 2. Set of recommendations for improving both the user side and service side

	User side	Service side
Metadata about data	Metadata about data in the SDI catalogue should appear in web search engine results (catalogue services should present all records as HTML pages).	Metadata should be available in ISO 19139, well formed and valid (conforming to the XML schema) XML documents (ISO 19139).
	Metadata should link to the distributor website, where data can be downloaded, and make restrictions of use and fees clear.	Metadata should link directly to the data itself. This can be a WFS, a WCS (Whiteside and Evans 2008) or a downloadable GML or raster file. Automatic metadata extraction should be used to synchronize metadata with data and decrease production effort (Olfat *et al.* 2010).
	Metadata should be produced and catalogued as close to data as possible (Rajabifard *et al.* 2009). Catalogues should be connected so that they can be interrogated together in a portal (Yang *et al.* 2006, Craglia *et al.* 2008).	Each catalogue should allow CSW requests and should return XML ISO 19139 metadata record sets. Metadata records should have a unique identifier among catalogues (Nebert *et al.* 2007).
	Portals should allow producers to create metadata records in the SDI catalogues (Goodchild *et al.* 2007). They should also inform the user about the degree of agreement to the SDI profile and provide quantitative measures of the quality of the metadata record that is being created.	There should be a mechanism for introducing an XML ISO 19139 record into an SDI catalogue (Goodchild 2007a). This system should evaluate the conformance to the SDI profile and reject inappropriate metadata records (Goodchild 2007a).
	Specific SDI metadata catalogues should allow users to search by words and regions, and keep advanced searching by metadata entries for experts. Graphical and visual statistical interfaces should help the user to formulate an advanced search request (Albertoni *et al.* 2004).	Catalogues should allow complex metadata queries based on different metadata entries using filter language.
	Catalogues should show a dataset series as a single metadata entity that can later be interrogated further (Manigas *et al.* 2009, Zabala and Masó 2005).	Applications should be able to select metadata of a series as a whole or as separate sheet datasets depending on how the data will be recovered (WFS continuous feature type (Vretanos 2005) or individual sheet download).

(*Continued*)

Table 2. (Continued)

	User side	Service side
Metadata about services	Descriptive metadata documents conformant to ISO 19119 should be available for each server in a human-readable form (ISO 19119). Metadata about servers in the SDI catalogue should appear in web search engine results.	ISO 19119 metadata should be linked to the corresponding service description file (WSDL or OGC capabilities document).
	Descriptive metadata documents that comprehensively describe the operations of the service should clearly specify data types for inputs and outputs.	An XML file should describe each operation (WPS DescribeProcess operation response (Schut 2007)).
	Service metadata should link to a portal or an application with an interface that allows the user to run the service.	Detailed technical descriptions of all the server's operations should be available to allow new applications to be built that can use this service or chain it to other services (Alameh 2003).
	The service catalogue should allow users to search by layer/feature type/coverage name over WMS, WFS, WCS and SOS services as well as allow users to follow links connecting to metadata about data.	Service metadata elements that allow the user to link to the data itself or to services for downloading should be filled in.
	Statistics of use and reliability information should be supplied to the user with service metadata query results (Wu *et al.* 2011).	Availability (Wang and Liu 2009) and conformance to the standard (Bernard *et al.* 2005b) should be tested for each service and collected in the catalogue from time to time.
Data models	ISO 19110 catalogues should collect the data models of feature datasets and present them in a human-readable format (Nebert 2004).	GML data should explicitly link to their application schema allowing validation (Manso-Callejo and Wachowicz 2009, Zimmermann *et al.* 2006).
	Data model HTML files should appear in web search engine results.	GML application schema should be available as a downloadable file or as a WFS DescribeFeature response (Vretanos 2005).
	Specific catalogues should collect data dictionaries in order to help harmonize concepts between datasets (Craglia *et al.* 2008, Wright *et al.* 2003).	Data dictionaries should be publicly available in GML format to be linked by GML feature collections.
		GML should be used to describe raster data models to make integration with feature data models easier.

(*Continued*)

Table 2. (Continued)

	User side	Service side
Data downloading	Data should to be available in a format that is easy to visualize and print (GeoPDF, KML, MMZ or JPEG).	Data should be available in an interoperable format (GML, GeoTIFF, etc.) as a complete file or filtered by a WFS or WCS server. Authentication or payment may be necessary in a cost recovery approach (van Loenen 2009).
		GML files should be compressed before network transfer due to the very verbose nature of GML (Yang *et al.* 2006).
	Simple feature collections should be distributed so that they can be easily represented on any modern device.	Complex features should be avoided and replaced by common GML profiles (Manso-Callejo and Wachowicz 2009). The GML simple features profile should be encouraged.
	Data should be distributed in the original format, which would help to preserve the original data model and all the properties recorded.	Interoperable formats should coexist with original formats. Some services should be able to access original formats. Transformation services should be able to transform from the original data model to the interoperable data model and format (Wright *et al.* 2003).
Data and processing services	The server URL should be visible in portals to allow users to open it from other portals and GIS applications.	Servers should be included in the server catalogue so that they can be discovered by applications and to allow service chaining (Tu *et al.* 2004).
	Web portals should present the information fast, and use techniques that create a perception of fast performance for the user (Tu *et al.* 2004).	Multi-threading (Yang *et al.* 2005), dynamic caching (Scholten *et al.* 2006) and binary compression (Yang *et al.* 2006) should be used to increase overall performance.
	Portals should always be available (24 × 7) and cannot change the URL frequently (Crompvoets *et al.* 2004).	Redundancy mechanisms and good scalable services should be used to guarantee the quality of service (QoS) (performance and reliability) and that the service is always online (24 × 7) (Yang *et al.* 2006).
	Map portals should incorporate postal addresses, requesting gazetteers, since they are a very popular way of referencing geospatial places.	Gazetteer services should be available for automatic conversion of a long list of addresses.

(*Continued*)

Table 2. (Continued)

	User side	Service side
	Portals should be easy to use but not too simplistic. New geoservices should add interfaces to execute geospatial processes and operations.	Services can be described in SOAP and WPS (Schut 2007). When possible, a WPS description of the service should be provided. A WPS capacities document should be included in the service metadata catalogue.
	Map browsers should be able to change projections. The whole world WGS84 projection and a projection without distortions on detailed views should be provided.	Fast coordinate transformation services should be available.
	Portals should show data from sensors (Craglia *et al.* 2008). This could be mainly point maps and interpolated maps (that can be understood more easily).	Sensor Observation Service (SOS) (Na and Priest 2007), Sensor Planning Service (SPS) (Dibner 2007) and other related Sensor Web Enablement (SWE) services should be catalogued and integrated with other services.
Portrayal and symbolization	Portals should provide several thematic presets of data to be chosen as a starting point (Craglia *et al.* 2008).	WMS portals and GIS applications should support Web Map Context (WMC) (Sonnet 2005) to save a preset and to interoperate with other WMS clients. The WMC document should be examined to discover other non-catalogued WMS servers.
	Map browsers should take care of historical releases of the same dataset and allow people to compare releases (Craglia *et al.* 2008, Wright *et al.* 2003).	A WMS, WMTS and WCS TIME parameter should be used when historical releases of the same dataset are available (Ostlander *et al.* 2005).
	Map browsers should chain different resolution products together in order to provide scale continuous data; otherwise users become frustrated or confused (Scholten *et al.* 2006).	WMS (de la Beaujardiere 2004) and WMTS (Masó *et al.* 2010) servers should provide data in a large range of scales, from the original resolution to very coarse resolutions (Bernard *et al.* 2005b).
	Portals should use a tiled approach for requesting maps (like WMTS). WMTS has been designed to respond better to pan and zoom actions on map browsers. It can redraw faster by using the Internet caching mechanism (Masó *et al.* 2009).	To increase performance, a pyramid index method should be used for images to serve WMS (Yang *et al.* 2006) (or WMTS). Binary compressed GML should be responded to by WFS (Tu *et al.* 2004, Yang *et al.* 2006).

(*Continued*)

Table 2. (Continued)

	User side	Service side
	The height and position of the current point should be included in map browsers.	A fast WMS GetFeatureInfo (de la Beaujardiere 2004) should be used to supply height (Z) from the DEM layer. The derived concepts (slope, aspect, etc.) can also be returned.
	Alphanumeric information related to the objects should be retrievable and presented in map browsers. Metadata about the data displayed should also be easily found.	WMS GetFeatureInfo should be supported for all layers that have alphanumeric information related to the objects or cells on the map (Vanmeulebrouk *et al.* 2009). Metadata about data should be linked to service metadata documents.
	Portals should be able to represent tabular information on a map, linking it to spatial objects. The new resulting dataset can be added to the catalogue (Wright *et al.* 2003).	The Table Joined Service (TJS) standard should be used to extend WMS and to allow new attribute data to be included in preexisting geospatial objects (Schut 2010). This will affect feature symbolization and enrich GetFeatureInfo responses.
	The description of the symbology should be available through an ISO 19117 portrayal conformance document (ISO 19117). Portrayal catalogue records should appear in web search engine results.	Symbology should be available as an OGC Symbology Encoding XML document (Muller 2006, Ostlander *et al.* 2005) or by a GML version 3.0 default symbology.
Mass market, VGI and Web 2.0	Users should be able to see SDI products on virtual globes (Gouveia and Fonseca 2008).	KML files should be provided with links to WMS services or a vector data representation.
	SDI should have participatory mechanisms that allow people to debate about products, evaluate data or metadata or even work together on georeferencing collective interests (Craglia *et al.* 2008).	An API should be provided to allow others to mash-up SDI resources into their web applications or portals (Chow 2008).
	User comments on metadata, quality and error detection should be collected in SDI portals (Craglia *et al.* 2008).	Web services should allow user comments to be recovered and should associate them with SDI resources.
	Users should be allowed to tag datasets and to use these tags for discovery (Kalantari *et al.* 2010).	Web services should save user tags and should associate them with SDI resources. Search engines should allow queries by tagging.

(*Continued*)

Table 2. (Continued)

	User side	Service side
	SDI portals should provide a participative environment in which datasets can be created collectively (Masser 2009).	WFS-T should be the base for collective creation of datasets in VGI environments.
	Statistics of use and questionnaires should help to improve catalogue responses.	Metadata and web services that are most frequently searched for should be counted and incorporated as search criteria.
	Presets of resource collections should be offered to the user.	User combinations of information should also be collected to improve further user interaction.
General	Users should have the perception that all the elements that compose a representation of a dataset are easy to collect.	All the elements should have a unique identifier, which should be used to link them together. Validation tools should test the correct presence of a dataset in the SDI: metadata, data, data model, services and symbology should be recorded and available. User-generated material should also be linked to the corresponding official dataset.

the harvesting process by limiting it to lower levels, but not all SDIs can be classified at a particular level of the SDI hierarchy (some good examples are academic and thematic SDIs). If records are duplicated in several catalogues, universal and unique metadata identifiers need to be used in order to recognize records and avoid reharvesting the same metadata record and filtering metadata more than once. Unfortunately, there is currently no consensus on how to generate and manage unique metadata identifiers, and the current version of ISO 19115 does not include this concept but rather only provides a data Uniform Resource Identifier (URI) (a common solution is to consider that metadata and data share the same identifier). Furthermore, there is no satisfactory way of making a bidirectional link between a dataset and its metadata record (Bernard *et al.* 2005b). This is particularly important when metadata are managed and updated because datasets change, especially when many producers store data and metadata in disconnected repositories (Olfat *et al.* 2010). This disconnection can result in metadata catalogues that do not contain all the available data, as we will see in the *Use Case 1* section, or contain metadata that are outdated or incomplete.

Despite the problems described, many issues can be addressed to increase metadata benefits. Even if the web search engines (like Google search) do not have geographic awareness, SDI metadata catalogues should allow these web search engines to index their metadata records. SDI visibility will be increased if metadata records appear in the web search engine results as human-readable documents. In addition, current metadata standards can link to many aspects of the data, such as a download URL to a file, a download service, a data model description, a legend or a symbolization encoding description, among others. These links are used very little as they are not part of the metadata core and have been defined as optional entries; however, they are important in the linked data domain (Goodwin *et al.* 2008) and on the web in general because they allow people to obtain and

use the data that have just been discovered. In fact, the absence of unique identifiers for metadata and the careless use of the previously mentioned links show that current SDIs are mainly based on the so-called service-oriented architectural style (sometimes referred to as remote procedure call based) instead of on a more resource-oriented architectural style (closer to how the Internet actually works) in which data, metadata and processes are available using URLs that can be retrieved and managed from a resource identifier (Mazzetti *et al.* 2009). Finally, a common problem is that metadata series tend to inflate the results of a metadata search, making it difficult to interpret the available datasets. In spite of the approaches to hierarchical metadata introduced in the ISO 19115 metadata standard (Annex H) (ISO 19115), there are only a few applications that correctly deal with dataset series, which indicates that more research on this issue is necessary (Zabala and Masó 2005).

Currently, metadata are mainly created manually in a monotonous, labour-intensive process that is viewed as an extra production cost. This results in an incomplete and irregular collection. New research shows that different methods of automatic metadata extraction (Han *et al.* 2003) can generate homogeneous metadata records that are well synchronized with the data from GML (Olfat *et al.* 2010) or image, digital terrain models and other vector formats (Manso-Callejo *et al.* 2009).

Some metadata entries, like keywords, can use a predefined value set that can be formalized in a dictionary of terms (thesaurus) containing at least one entry identifier and a definition for each term. Some SDI implementations have disregarded the importance of these dictionaries although they are a key issue for future semantic harmonization. Fortunately, the directive 2007/2/EC of the European parliament and of the council of 14 March 2007 to establish an Infrastructure for Spatial Information in the European Community (INSPIRE) is recommending the use of the general multilingual environmental thesaurus (Bernard *et al.* 2005b). Semantic interoperability is still under research and is currently difficult to achieve, but it should be encouraged by promoting these dictionaries.

Improving metadata about services

Metadata about services is also an open issue. The basic concepts are considered in ISO 19119 and OWS Common, and there are at least three ways to encode the description of a service:

- An ISO 19119 metadata description document that can be applied to all geospatial web services, including non-OGC (Crompvoets *et al.* 2004).
- OGC Web Services Common is characterized by a 'service metadata document' (previously known as a 'capabilities document'). This applies to the OGC standards for web maps (WMS) (de la Beaujardiere 2004), web tiled maps (WMTS) (Masó *et al.* 2010), features (WFS) (Vretanos 2005), coverages (WCS) (Whiteside and Evans 2008), services (e.g. the generic web processing services, WPS) (Ostlander *et al.* 2005) and to any other standard that follows OWS Common (Whiteside and Greenwood 2010).
- A Web Service Definition Language (WSDL) document that applies to all Simple Object Access Protocol (SOAP) web services (including the non-geospatial ones) (Alameh 2003).

An example of the XML fragment for these three encodings is shown in Figure 3. WSDL is only intended for machine to machine communication and does not contain most of the metadata elements needed for service discovery. OGC Web Services Common data

(a)

```
<MD_Metadata>
  <identificationInfo>
    <srv:SV_ServiceIdentification>
      <srv:containsOperations>
        <srv:SV_OperationMetadata>
          <srv:operationName>
            <gco:CharacterString>GetMap</gco:CharacterString>
          </srv:operationName>
          <srv:DCP>
            <srv:DCPList>KVP</srv:DCPList>
          </srv:DCP>
          <srv:connectPoint>
            <CI_OnlineResource>
              <linkage>
                <URL>http://www.miramon.uab.cat/cgi-bin/MiraMon.cgi?</URL>
              </linkage>
            </CI_OnlineResource>
          </srv:connectPoint>
        </srv:SV_OperationMetadata>
      </srv:containsOperations>
    </srv:SV_ServiceIdentification>
  </identificationInfo>
</MD_Metadata>
```

(b)

```
<Capabilities>
  <ows:OperationsMetadata>
    <ows:Operation name="GetMap">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:href="http://www.miramon.uab.cat/cgi-bin/MiraMon.cgi?"/>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="Format">
        <ows:AllowedValues>
          <ows:Value>image/jpeg</ows:Value>
        </ows:AllowedValues>
      </ows:Parameter>
      <ows:Parameter name="ExceptionFormat">
        <ows:AllowedValues>
          <ows:Value>text/xml</ows:Value>
          <ows:Value>application/vnd.ogc.se_inimage</ows:Value>
        </ows:AllowedValues>
      </ows:Parameter>
    </ows:Operation>
  </ows:OperationsMetadata>
</Capabilities>
```

(c)

```
<definitions>
  <binding name="WMS_HTTP_GET_Binding"
           type="wms_wsdl:WMS_HTTP_Port_GET">
    <http:binding verb="GET"/>
    <operation name="GetMap">
      <http:operation location=""/>
      <input>
        <http:urlEncoded/>
      </input>
      <output>
        <mime:content type="image/*"/>
      </output>
      <fault name="exception">
        <mime:mimeXML/>
      </fault>
    </operation>
  </binding>
</definitions>
```

Figure 3. Fragment of a service metadata XML document following: (a) ISO 19119, (b) OWS Common 1.2 and (c) WSDL.

model is based on ISO 19119 and it is only applicable to OGC services. For those reasons, the ISO 19119 is the most generic solution, so this option should be used in service metadata catalogues (even if it is quite verbose), but each record should still link to the actual OGC service metadata document and/or the WSDL document when they are available.

ISO 19119 covers the essential description of the service and its operations; however, some extra information on the service is not covered by this standard and should also be collected. Information characterizing the availability and reliability of the service can be obtained by testing the service from time to time and the results should be stored with the service metadata as reliability statistics. In addition, service usage metrics can also be collected and added to metadata (Wang and Liu 2009). Even the conformance to the standard or a profile can be tested using specific tools like the OGC testing tools developed in the CITE program (Bernard *et al.* 2005b). All this relevant information helps users to decide which service they prefer and can also be used by the catalogue search engine as criteria for organizing the search results.

Improving data models

The need to describe the data model (conceptually covered by the ISO 19109) of a dataset is frequently ignored in the GIS community and largely confused with data formats. The Federal Geographic Data Committee metadata standard (CSDGM) (Metadata Ad Hoc Working Group 1998) includes the feature and attribute description as part of the metadata documents; however, the more recent ISO 19115 metadata standard (ISO 19115) does not address this topic because it relies on a separate standard for feature catalogues (ISO 19110). SDI catalogues rarely include the data model because this information, although maybe not consciously, is not considered important for data discovery. This has led us to the undesirable situation in which producers do not share information about the data model easily.

Apart from CSDGM, there are two generic ways of providing standardized data model descriptions:

- The ISO 19110 standard specifies how to describe feature types and feature attributes of the dataset (as well as operations and associations). The standard does not include any encoding, but the ISO XML team provides a way of describing it as an XML file (Nebert 2004).
- The ISO 19136 standard provides a set of rules for describing the feature types and feature attributes (GML properties) that are represented in the dataset as a GML application schema document (XSD document) (Manso-Callejo and Wachowicz 2009).

The two alternative data model descriptions complement each other and should not be seen as different alternatives: ISO 19110 contains a more human-readable description of the feature and attribute types in the dataset and their relations (even if this information is encoded in XML it can be easily converted to a readable HTML file), whereas OGC-GML is a machine-readable XSD document that defines GML feature and attribute types, including the information needed to guarantee only XML validation.

Tools to develop and transform data models are necessary, even if these transformations require human intervention to complete the work (like the Snowflake software). Examples of possible semiautomatic transformations are from GML schemas to ISO 19110; from

ISO 19110 XML documents to GML schemas; and from implicit data models in formats like SHP or DXF to GML application schemas or to ISO 19110 XML documents. More research and development should be carried out on data model transformations and cataloguing these models.

Some feature properties contain categorical information that can take a set of predefined values. Category dictionaries should also be provided in some form, for example, they could be encoded in GML and linked to GML application schemas and instances.

Improving data download

The main problem with data is that the current SDI implementations are clearly focused on portrayal and map representation and do not provide enough information on the service side (or it is difficult to find) to allow applications and services to obtain and analyse the data. Today professional GIS tools are ready to connect to the SDI web services and use their data (Maguire and Longley 2005). In fact, many professional GIS desktop software (like ArcGIS, Autodesk MapGuide, MiraMon, etc.) provide implementations of OGC-WMS map clients that can connect to distributed WMS servers, but these connections are useless for spatial analysis since they only provide graphical representations of the data. Spatial analytical tools require access to the data itself, but there are still few real data web services (e.g. WFS and WCS).

In first-generation SDIs all data were available in the SDI portal and were easy to obtain if the user had access rights. Second-generation SDIs aim to create distributed data systems that make it 'appear as if the information in the component systems was stored and maintained in a single centralized database'. In other words, to integrate data without centralization, a logical rather than a physical integration (Xue *et al.* 2002) has been chosen. However, for practical reasons, second-generation SDIs have partially failed to achieve this aim and have involuntarily introduced confusion into the availability of the data due to at least three factors:

- The availability of metadata records seems to be enough. As we explained above, metadata and data are rarely linked together; therefore, accessing a metadata record does not necessarily mean easily accessing the data itself. In many cases, the user has to search again at the producer's website or contact them directly.
- The GIS industry has implemented WMS connectors that are integrated into the SDI geoportal website and provide freely available data visualization (Bernard *et al.* 2005b) without providing download capabilities by any means. Nonprofessional users may believe, when looking at the WMS 'images', that they have data to work with if necessary. Another example of a recent portrayal solution is GeoPDF (Cervantes 2009).
- When providers make the effort to provide the real data, they prefer a standard way of doing this. Two scenarios can arise: the provider that prefers a *de facto* standard data format (like GeoTIFF, SHP or DXF) and the one that chooses a *de jure* standard. In the first case, superior usability is often achieved, but there are the corresponding drawbacks related to older formats (no explicit data model, failure to support new features such as hyperlinks or connections to database management systems). Conversely, the provider may choose, for example, to provide vector data with the OGC Web Feature Service standard. The WFS uses GML as a neutral file format. In theory, the GML file is very flexible and allows complex geometries and rich attributes. A GML file can also be read by some GIS tools. In practice, only a few

> software applications can read GML, although in some cases they lose alphanumeric attributes or topology information. There is a general misunderstanding and many people believe that WFS only allows a response to be made in GML, but in fact the WFS standard allows any format and the only requirement is to provide a GML application schema of the data model.

For these reasons, the second-generation SDIs have orientated the infrastructure to a situation that makes it difficult for users to obtain data directly from the SDI portals. The users become frustrated and are forced to look up the original providers. In fact, Vandenbroucke (2008) demonstrates that in Europe the very same countries that have complete metadata collections and map viewers only have a small percentage of them available for download (except for Austria and Norway). Crompvoets *et al.* (2004) suggested that, in recent years, the use of SDI facilities is not growing as fast as expected in several aspects.

Given the current situation, we believe that SDI implementations should return to the original aim: to renew interest in true data. Metadata records should link to the original data in the original format. Sometimes, the original format was chosen for collection properties and relations between the elements that other formats cannot express or express in a more indirect way. It could be argued that this could be less interoperable than a standard format (e.g. GML obtained through a WFS). This issue needs to be further addressed (Tait 2005), but the real situation is not so bad. There are a few formats that have become *de facto* standards (SHP, etc.) and there is a very complete collection of data transformation tools (among these *de facto* standards) that work reasonably well, some of which are freely available, even in open source (e.g. http://www.gdal.org/).

Improving data services and adding processing services

Some authors consider that the geoportals that show data in the 'old fashioned' two-dimensional layers are unattractive for the general public, who prefer virtual globes (Craglia *et al.* 2008), especially Google Earth (Butler 2006). There are also other reasons for the success of virtual globes, such as the easier way of finding postal addresses or the embedded access to better and faster search engine technologies. This is the key factor of the Google Search success over other competitors. We believe it is also the key factor in the success of virtual globes in general, and Google Earth in particular, over SDI geoportals in the mass market arena.

The reliability and performance of web servers are serious business (Bernard *et al.* 2005). Once servers are in place, people may start using them frequently; service implementations should scale in a way that, if the demand increases, they can maintain the performance level. Publishing 24×7 content reliably is neither simple nor inexpensive (Tait 2005). In fact, spatial web services need to support a large number of concurrent requests, unleashing complex computations and requiring large-quantity data transmission (Tu *et al.* 2004). Surprisingly, only a few papers deal with the reliability and performance of SDI web services (Tu *et al.* 2004, Yang *et al.* 2005, 2006, Scholten *et al.* 2006). Sometimes it is impossible to guarantee the quality of the service and the scalability of the design with a single computer, and a cluster of computers should be considered (Yang *et al.* 2005).

On the client side, clients have to be able to request data asynchronously in a multi-threaded way, allowing users to continue interacting with the client instead of waiting for the previous request to get back. It is difficult to find the equilibrium between requesting only the part of the data the user really needs (Scholten *et al.* 2006), or reducing the number of concurrent requests that the server will receive, and anticipating user data needs by

transmitting larger pieces of information. However, the number of server requests can be reduced by using caching mechanisms in map and tile servers (Yang *et al.* 2005) or by avoiding requesting single features and conveniently packing requests for feature collections (requesting features in a bounding box of several feature classes, etc.) in a WFS server (Tu *et al.* 2004). In the last case, features have to be retrieved as a binary compressed GML (Tu *et al.* 2004, Yang *et al.* 2006). However, sometimes it is considered more important to reduce system response latency and increase interactivity with the user by fragmenting a request into small pieces of information (increasing granularity and the number of individual requests), even if the sum of times of the granular responses is larger than the response of a request that retrieves all the information at once.

From a technological point of view, the recent great success of web asynchronous data access technologies, such as web browsers based on asynchronous JavaScript and XML and asynchronous JavaScript Object Notation, can be applied to reduce latency. They can also be useful for distributed geoprocessing in which large volumes of data can be accessed progressively (Zimmermann *et al.* 2006). Developers of servers that respond to JavaScript and XML and JavaScript Object Notation clients should concentrate their efforts on supporting many small concurrent requests.

Several papers on distributed geoprocessing are emerging that demonstrate use cases in which geoprocessing is technologically possible (Xue *et al.* 2002, Scholten *et al.* 2006). Recently, a special issue of the *International Journal of Geographic Information Science* was dedicated to distributed geoprocessing (Yang and Raskin 2009). Some authors even state that current web services are in a preliminary step rather than at a final solution, foreseeing models and geoprocessing as a necessary next step (Wright *et al.* 2003). On the other hand, the emerging concept of cloud computing is seen as an opportunity to expand distributed geoprocessing. Primary SDI uses of geoprocesses should be coordinate transformation or format transformation services, but in fact WPS (Schut 2007) could encapsulate interfaces for almost every geospatial analytical process. Using WPS for geoprocessing should be considered as it would complement the generic WSDL-SOAP used in software development communities.

Communities change over time, and therefore the ideal SDI should expand accordingly (Kok and Loenen 2005). Some SDI initiatives around the globe are meeting resistance in some fields, such as meteorology, hydrology and traffic monitoring, which depend greatly on information that comes for sensors near, on or over the earth's surface. These fields measure physical characteristics (pressure, temperature, humidity) and phenomena (wind, rain, earthquakes) or keep track of animals, vehicles or people with wired or wireless sensor networks (Craglia *et al.* 2008). OGC has released a set of standards under the generic name of Sensor Web Enablement, and particularly OGC Sensor Observation Service (Na and Priest 2007) should be adopted by SDI. Geosensor data are supplied in real time or near real time usually with measures every few minutes; therefore, information is fragmented into small pieces making it different from the traditional static maps provided by NMA (Craglia *et al.* 2008). To harmonize it to NMA maps, data distribution models, interpolation and dynamic systems should be developed and served with raw sensor data.

Improving data portrayal and symbolization

Similar data at different scales from the same institution (e.g. 1:5000, 1:25,000, etc., topographic maps) or at the same scale but coming from different sources (e.g. 1:25,000 topographic maps at either side of national borders and produced by the respective NMA

sharing the border) often use heterogeneous symbolization. The first case may be justified due to different entity and symbolization needs at each scale, but the second case is especially difficult to maintain even if those countries have spatial data agreements (as is the case of INSPIRE in the European Union). Indeed, to be able to meaningfully integrate spatial data from different sources, a uniform representation of spatial objects is required; however, often organizations are not willing to modify the presentation (or facilitate modification of the presentation by a third party) of spatial data to fit another purpose. Therefore, most service implementations do not have this functionality (Vandenbroucke 2008).

The symbolization should be described using ISO 19117 (ISO 19117) or OGC Symbology Encoding (Muller 2006) and should be applied to a WMS service through styled layer descriptors (Muller and MacGill 2005). To prevent heterogeneous symbolization, SDIs should have portrayal catalogues in which Symbology Encoding documents can be stored and reused. New datasets could reuse the same symbolization, especially if they describe the same feature type. Efforts to harmonize ISO 19117 and OGC Symbology Encoding are currently underway in OGC interoperability programs. There is also default symbolization encoding in GML version 3 but it is the one used in GML documents.

WMS and WMTS web portals should be simple and easy to use but not too simplistic. Web map browsers in SDI portals should provide several thematic presets to show the thematic diversity of the available data. OGC Web Map Context (Sonnet 2005) is an excellent technological resource for defining presets because it defines a way of saving and retrieving WMS layer combinations. In addition, web portals should provide a way of managing time series. The WMS and WMTS GetFeatureInfo or GetFeature operations should be used to show the alphanumeric data behind maps (Vanmeulebrouk *et al.* 2009).

Adding mass market, VGI and Web 2.0

Second-generation SDIs repeat a model in which producers suppose that the product satisfies the user needs and users will employ these products in the way the producer intended (Budhathoki *et al.* 2008). Users have almost no way of contributing to the SDI content.

SDIs target professional map users, but they should take advantage of mass market tools and access a broader audience. Virtual globes do not currently have highly diverse, quality thematic content, like SDIs do; therefore, thematic data providers integrated in SDIs should take advantage of virtual globes to target non-GIS users more easily (Gouveia and Fonseca 2008). WMS and WMTS servers can already be loaded in virtual globes, but automatically generated KML or similar files should be provided to make accessing SDI services easier for the public. They could also include a vector representation of the data like in Google Earth.

VGI offers enormous opportunities for developing SDIs, but the potential of VGI has not yet been intensively exploited or even fully understood. Indeed, current citizen participation is limited to isolated efforts and *ad hoc* initiatives (Gouveia and Fonseca 2008), some of which are clearly successful. Unofficial amateur groups with a common interest and the right set of tools can produce, for example, weather data (the GLOBE program, http://www.globe.gov) (Goodchild 2007b) or ornithology datasets (http://ebird.org/content/ebird). Users can collect information themselves or obtain it by sensors (like an amateur meteorological station or a handheld GPS for country walks). The usefulness of these heterogeneous datasets raises issues of quality and the lack of metadata that need to be studied further (Gouveia and Fonseca 2008). Internet Web 2.0 and social network technologies should be used to collectively create and exchange information.

On the other hand, a clear framework for more constant user involvement should be set. Crompvoets *et al.* (2004) suggested that standards are difficult to understand and use languages that are too formal (this is especially true for metadata standards). This should be solved by allowing users to comment on metadata records, combining the formal data and the informal user contributions in the same reading environment. A user would need to identify itself in the system before it could submit comments. This could be seen as an initial obstacle because apparently people do not like to fill in registration forms. Unexpectedly, with the data collected so far, Crompvoets *et al.* (2004) demonstrated that SDIs that require authentication have the same amount of users as the openly available ones. In addition, users can tag datasets directly and use them in searches. User tags have no semantics associated with them and express user vocabularies (sometimes referred to as folksonomies) rather than keywords, with which producers express the classification criteria by picking terms from controlled vocabularies (Kalantari *et al.* 2010). Even though tags are occasionally ambiguous, they provide a user perspective and result in more user-oriented metadata. The combination of formal and informal metadata in a Web 2.0 environment is illustrated in a pilot project in the next section.

The number of hits is a parameter generally collected in geoservers but user dynamics generate richer information that should be collected: user behaviour and preferences in geoportals. Once collected, this information should be used to improve search engine results by introducing data on favourite datasets. Furthermore, layer combinations in the geoportal website are a source of useful information and should be used to configure future general favourite presets.

We have presented the points we need to address to improve the current-generation SDI. The following sections of this article provide two use cases in which we apply some of the above-mentioned recommendations. The first use case evaluates the accessibility to healthcare centres in Catalonia and the second use case presents a Web 2.0 implementation in which users can complement metadata records.

Use case 1: accessibility to healthcare centres

The Catalan SDI, IDEC, is a fine example of an advanced SDI in Europe (Craglia and Campagna 2009). It is considered in several research papers as a good representative of a sub-national SDI (Masser *et al.* 2008, Nedović-Budić *et al.* 2008, Donker 2009, Paudyal *et al.* 2009, Rajabifard *et al.* 2009). It has a catalogue with 37,447 metadata records from 130 organisms, such as sub-national and local administrations, the Catalan cartographic agency (ICC), different universities (especially the UAB) and the private sector (see January 2011 statistics at its website). Although about half of the records come from the different sheets of cartographic series (e.g. 1:5000 and 1:25,000 orthophotos and topographic maps) and different dates of satellite images, this figure is huge for a country of about 32,000 km^2. One of the first ISO 19115 metadata editor applications (MetaD) was developed in the first stage of the Catalan SDI (Rajabifard *et al.* 2009). The infrastructure has a WMS visualization geoportal with some data that can be downloaded using WCS/WFS protocols or in HTTP links to GIS file formats. It also has some fine examples of SOAP-WSDL services.

Health is one of the potential benefit areas of geospatial data and GIS technologies. Nevertheless, this area has rarely been represented in generic SDI catalogues in the past, and only recently have the benefits of harmonization and data sharing become of interest for this sector (Thompson *et al.* 2009). Unsurprisingly, there are only a few records directly related to health in the Catalan SDI database.

Here, we present our experience in a relatively straightforward use case that involved collecting data, executing some analytical algorithms and getting back some results, all within the context of the Catalan SDI. The aim of our use case was to use SDI tools and GIS software to analyse the geographic accessibility of population settlements to healthcare centres in the Catalan public health system. In our geographic accessibility study, the source is the population settlements, the target is healthcare centres and the medium is the road network. In this analysis, we needed to locate georeferenced healthcare centres and make calculations on travel times and distances from the population settlements to the healthcare services. This is not a fictional application case, but rather a real one that was carried out in 2006–2007 (Olivet *et al.* 2008) and which was an important cartographic database for the public investment plan approved in 2008 by the Catalan Parliament. It is important to note that all the following comments correspond to the catalogue status at the time the study was carried out, but it is assumed that nowadays (2011) the situation is not better in most other SDIs. This use case shows several aspects that were discussed in previous sections of this article.

The use case started by collecting the necessary information: georeferenced healthcare centres (points), population settlements (points) and a road network (topological line graph). In the Catalan SDI, we found that there were no datasets from the Catalan Health Department in the catalogue. Only a few municipalities had entered data about basic health areas (the basic healthcare administrative unit). The Catalan Health Department provided this information directly as a postal address database. The Catalan SDI has a *gazetteer* as a web gadget that centres the view of a map browser on a particular address, but there is no documented way of using this gadget as a web service (i.e. a way to automatically get the coordinates of the 1546 record database).

The Catalan SDI did not have a population settlement dataset metadata record. We discovered that the government department had a useful point dataset that fitted our purpose better, so we asked them directly. The Catalan SDI metadata catalogue did not contain any records from the government department. The Territorial Policy and Public Works Department provided us with a useful road graph with average travel speeds as one of the attributes. Metadata or data about this layer were not in the Catalan SDI catalogue.

There is a small set of geoprocessing services available on this SDI. The service catalogue has grown to 229 records that come from 66 organizations. It contains mainly WMS, WFS and WCS services, and only 11 records for WPS or SOAP-WSDL geoservices (4 August 2009 statistics on the Catalan SDI website). The 11 available WPS services were not applicable at all to what we needed, which is the general case in the SDI domain (Craglia *et al.* 2008). Therefore, instead of using a geoservice, we chose the MiraMon GIS software (Pons 2004) to do the analytical work.

Finally, isochrone maps were produced reflecting the minimum distance and minimum time needed to reach a specific kind of healthcare service (see Figure 4). The MiraMon software has an interesting way of dealing with metadata because the software keeps track of all the processes and builds the metadata record for a new derived layer using metadata from the previous layers and the information from the process itself. The producer has to edit a few extra things (that cannot be deduced automatically) to complete the record (this can be carried out with the ISO metadata manager (GeMM) the software has provided since 2001: Zabala and Pons 2002). Some papers from the recent literature argue in favour of this approach (Rajabifard *et al.* 2009). Then you can generate the ISO 19139 XML metadata-compliant document automatically, but the only way to submit a metadata document to the Catalan SDI is to download the MetaD metadata editor and manually fill the metadata records in one by one, field by field and send the result to the catalogue (using an option

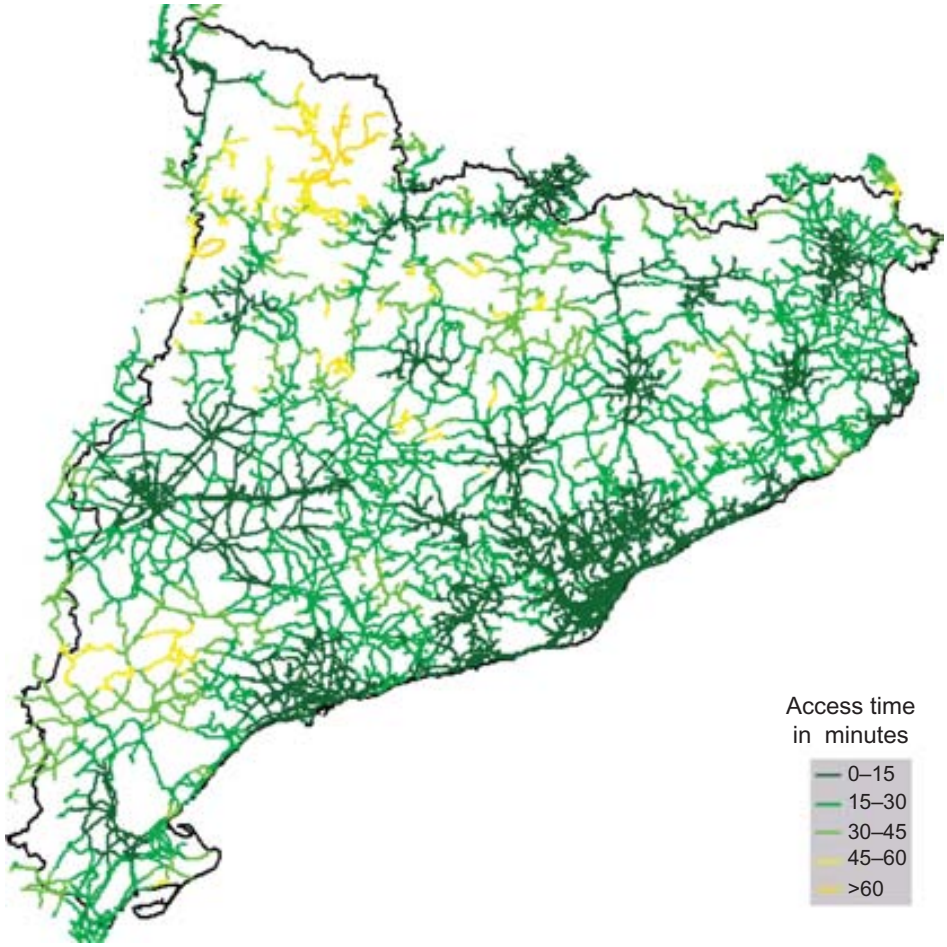


Figure 4. Isochrone dataset reflecting the number of minutes that a person needs to travel from their nearest road to the nearest drug addiction attention centre. The colours yellow and amber indicate more than 45 minutes.

in the menu). We asked IDEC if there was any other way, and they allowed us to send the metadata XML documents exported by the MiraMon software, and therefore we could skip the process of retyping everything on the MetaD tool.

In summary, the study on accessibility to healthcare centres in the Catalan public health system using SDI portal tools revealed some aspects that need to be improved:

- As we expose in the section *Improving data services and adding processing services*, the service side should be considered in SDI implementations not only for visualization and portrayal but for all web tools, making them available as web services using an international standardized protocol and providing a URL entry point. In our use case some tools were available to users as web forms (e.g. *gazetteer*) but there was no way of using them as web services.
- As we discuss in the section *Adding Mass Market*, *VGI and Web 2.0*, Web 2.0 and VGI tools need to be included in SDI implementation to allow users to report on

data that are not in the catalogue or to improve their availability. In our use case, the Catalan SDI catalogue did not show the geoinformation we needed, which did actually exist in the Catalan administration departments. There was no problem with the metadata catalogue search engine but rather in the completeness of the catalogue. If there are reports on missing data other users can check them and producers can correct the situation.

- Connecting to the section *Improving metadata about data*, producers should be allowed to include their products easily in the SDI. In our use case, the Catalan SDI website should clarify how to publish metadata in their catalogue. MetaD is the only way of publishing metadata about data and there is no way of reporting a service.
- Again about *Improving metadata about data*, the method of calling the right people in the right department of the Catalan administration on the phone and asking them for what we needed gave us more results than the SDI metadata catalogue. Goodchild *et al.* (2007) describe the process in which potential users with a request for a dataset adapt their needs to the actual data available by talking with experts and being advised by them in a *cognitive issue*. This process is difficult to emulate in the current metadata search engines and more research is necessary. Perhaps, a solution could be found by exploring the graphical approach to searches presented in Albertoni *et al.* (2004).
- As we explain in the section *Improving data services and adding processing services*, many SDIs have very limited geoprocessing capabilities. The Catalan SDI is not an exception and should promote the creation of new geoprocessing services.

It is important to note that to solve the above-listed problems, a reformulation of the SDI concept is not required, but rather better use of the available resources. In addition, these problems are not exclusive to the Catalan SDI and our experience suggests that similar use cases could be tested within other national and sub-national SDI with similar results. We chose this particular SDI because, as mentioned above, it is generally agreed that it is one of the earliest and most complete SDIs and because a critical approach can be carried out best with data and administrations close to our experience.

Use case 2: Web 2.0 user metadata comments: IDECTalk

Current standard-based approaches to metadata capture require considerable human input and are difficult to keep up to date. They primarily represent the perspective of the data producer (on quality and utility of the data). Since quality is currently defined as fitness-for-purpose, user feedback is essential for expressing the users' measures and opinions about the dataset. In addition, metadata is distributed separately from the data themselves (Craglia *et al.* 2008).

Some of the current limitations of metadata could be overcome with more participative methods of user classification and feedback, as is already a common practice in commercial Web 2.0 services. This use case puts into practice some aspects previously discussed in the *Adding mass market, VGI and Web 2.0* section.

Some papers explore the possibility of using VGI as an alternative to the official sources and suggest that it is possible to apply some of the Web 2.0 methodologies to the data or metadata collection (Goodchild 2007b, Gouveia and Fonseca 2008, Kalantari *et al.* 2010). These papers define different aspects of a possible framework but do not provide practical developments or a pilot project.

Here, we present a pilot project which allows VGI about metadata to be created and demonstrate that some of these problems (that also affect the Catalan SDI) can be overcome with a search portal connected to the SDI metadata catalogue. Indeed, this portal allows metadata about a particular topic to be requested and a particular dataset identified. The user can read the catalogued metadata from the producer and also the previous user comments. Moreover, they can enter new comments or update their own. All of this is stored and becomes immediately available to other users.

The presented environment is a mash-up that relies on the Catalan SDI catalogue and mass market mapping tools. It is a CGI application connected to a database that stores user comments and that resides in a web server. We use universal and unique identifiers that the Catalan SDI assigns to each metadata record to link both official metadata records and user comments. Figure 5 shows the relationship between all of these elements.

A user session is initiated with a search for a work or sentence. The results come back in the same way that they do in the Catalan catalogue web page, but they also include information on the previous user comments (Figure 6).

If the user selects a metadata record, it is shown on the screen (Figure 7). The dataset bounding box extracted from the metadata record is also shown using the Google Maps API, as another mash-up (Chow 2008). Previous user comments are also shown and it is possible to add new comments. To introduce new comments, user authentication is needed. Many metadata records include the producer contact information. When the producer's email address is available, new comments about a particular dataset are immediately emailed to them, so the producer can take action.

This experimental environment was developed in less than a week and no direct communication with the SDI support centre was required. This demonstrates that setting up Web 2.0 applications that consider users' comments connected to an SDI metadata catalogue can be easily deployed by any third party, who immediately becomes part of the SDI infrastructure. Although the original intention was to demonstrate that a third party can easily contribute to the SDI using metadata identifiers as links to the SDI catalogue, further conversations with the people in charge of the Catalan SDI led us to the conclusion that the usability of the platform will be increased if it is incorporated into the Catalan SDI portal and search engine, which they are willing to do in the near feature. The

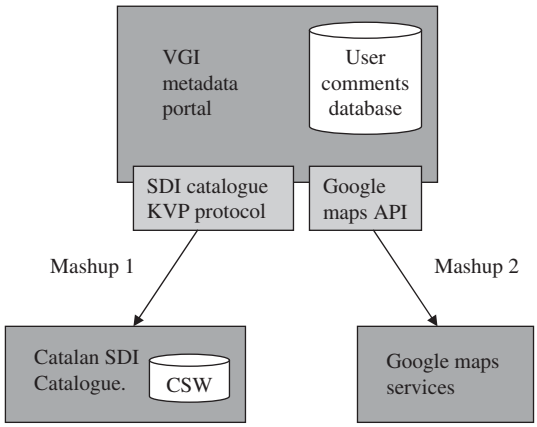


Figure 5. IDECTalk architecture.

Figure 6. IDECTalk search results with some metadata records that have previous user comments.

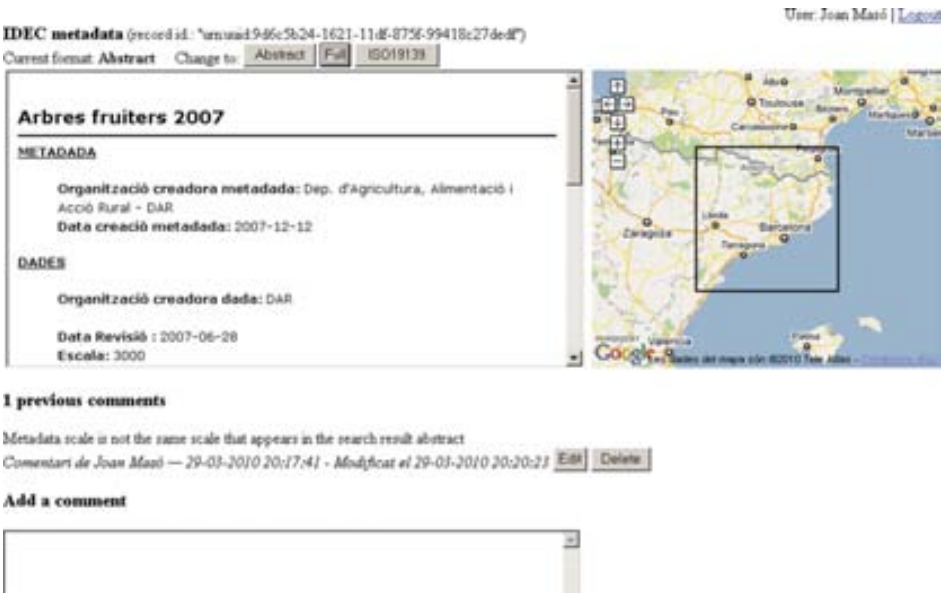


Figure 7. IDECTalk metadata record with previous user comments and the possibility of adding new ones.

experimental environment deliberately uses a style that is similar to web search engines and blog applications so the user feels immediately familiar with the system.

Clearly, there are some issues that need to be addressed more fully, like the real motivations that make people volunteer information and the process needed to ensure the quality

of the information provided (Craglia *et al.* 2008); however, we believe that the experiment is interesting as a starting point. Future lines of extension of this platform are the introduction of user tagging capabilities and user clouds (Kalantari *et al.* 2010) and to collect data about the user behaviour and adding this information to the system. This can be done in such a way that users can also use this information, such as in a statistical form or as recommendations like 'users that search for . . . also search for . . .' as in other Web 2.0 services.

Conclusions

The SDI phenomenon has been growing continuously over the past two decades. A decentralized model powered by the adoption of international standards and Internet technologies has been crucial for deploying the second-generation SDI. Nevertheless, there are several issues in SDI development that need to be solved. A relatively straightforward practical use case in which the Catalan SDI was used to study accessibility to healthcare centres showed how the SDI failed both to provide information about the data available in the Catalan administration and to provide distributed analytical tools. This article demonstrates that for an SDI to be useful there is no immediate need for reconceptualizing its principles, but rather it is necessary to reinforce or even refocus some aspects of the current-generation SDIs. In table form, we list the performance and usability aspects that can be improved in order to obtain a good balance between client portal gadgets and a server application behind them that can make tasks easier for the actors currently involved in SDI development or even engage other collectives. The collection of metadata and the search processes need to be improved. Service metadata have to be clarified so that data as well as tools that work in a distributed environment can be found easily. True data availability with default symbolization needs to be combined with standard web services in a seamless environment that can be enriched with VGI. To demonstrate how easily VGI can help SDI development, we provide the example of a project website that mixes classical standard protocols with modern web mash-up techniques to allow volunteers to complete metadata catalogues with the user perspective, and therefore fill in the provider information gaps.

Acknowledgements

This research was carried out as part of the activities promoted by the Interoperability Framework for the Catalan Cartographic Plan and funded by the Institut Cartogràfic de Catalunya. This article has been written thanks to the support of the European Commission through the FP7-265178-GeoViQua (ENV.2010.4.1.2-2) and a grant to Consolidated Research Groups given by the Catalan Government (2009 SGR 1511). Xavier Pons is recipient of an ICREA Acadèmia Excellence in Research grant.

References

Alameh, N., 2003. Chaining geographic information web services. *IEEE Internet Computing*, 6 (18), 22–29.

Albertoni, R., Bertone, A., and Martino, M., 2004, Visual analysis of geographic metadata in a spatial data infrastructure. *In*: *Proceedings of the 15th international workshop on database and expert systems applications* (*DEXA'04*) Saragossa, Spain, August/September 2004. IEEE Press.

Beaumont, P., Longley, P.A., and Maguire, D.J., 2005. Geographic information portals—a UK perspective. *Computers, Environment and Urban Systems*, 29 (2), 49–69.

Bernard, L., et al. 2005a. Towards an SDI research agenda. *In*: *11 EC-GIS*, 28 June–1 July 2005 Sardinia, Italy.

Bernard, L., et al., 2005b. The European geoportal—one step towards the establishment of a European spatial data infrastructure. *Computers, Environment and Urban Systems*, 29 (1), 15–31.

Bregt, A. and Crompvoets, J., 2004. Spatial data infrastructures: hype or hit. *In*: *11 EC-GIS*, 28 June–1 July 2005 Sardinia, Italy.

Budhathoki, N.R., Bruce, B.C., and Nedovic-Budic, Z., 2008. Reconceptualizing the role of the user of spatial data infrastructure. *GeoJournal*, 72, 149–160.

Butler, D., 2006. Virtual globes – the web wide world. *Nature*, 439, 776–778.

Cervantes, D., 2009, *Using GIS to create an interactive GeoPDF mapbook for the Big Island of Hawaii* [online]. Thesis (PhD). Available from: http://www.nwmissouri.edu/library/theses/CervantesDanielle/FINAL_THESIS.pdf [Accessed 10 January 2011].

Chan, T.O., et al., 2001. The dynamic nature of spatial data infrastructure: a method of descriptive classification. *Geomatica*, 55 (1), 1–18.

Chow, T.E., 2008. The potential of maps APIs for internet GIS applications. *Transactions in GIS*, 12 (2), 179–191.

Coleman, D. and McLaulghlin, J.D., 1997. Information access and network usage in the emerging spatial information marketplace. *Journal of Urban and Regional Information Systems Association*, 9 (1), 8–19.

Craglia, M., et al., 2008. Next-generation digital earth. A position paper from the Vespucci Initiative for the advancement of geographic information science. *International Journal of Spatial Data Infrastructures Research*, 3, 146–167.

Craglia, M. and Campagna, M. 2009, Advanced regional spatial data infrastructures in Europe. Ispra, Italy: European Commission, Joint Research Centre, Institute for Environment and Sustainability.

Crompvoets, J. and Bregt, A., 2003. World status of national spatial data clearinghouses. *URISA Journal*, 15 (1), 43–50.

Crompvoets, J., et al., 2004. Assessing the worldwide developments of national spatial data clearinghouses. *International Journal of Geographical Information Science*, 18 (7), 665–689.

de la Beaujardiere, J., 2004. *OGC web map service (WMS) interface, version 1.3.0* [online], OGC 03-109r1. Available from: http://portal.opengis.org/files/?artifact_id=5316 [Accessed 20 March 2010].

Dibner P.C., 2007. OpenGIS® Sensor Planning Service (SOS) Implementation Specification, version 1.0.0 [online], OGC 07-014r3. Available from: http://portal.opengeospatial.org/files/?artifact_id=23180. [Accessed 7 January 2011].

Donker, F.W., 2009. Public sector geo web services: which business model will pay for a free lunch?*In*: B. Loenen, J.W.J. Besemer, and J.A. Zevenbergen, eds. *SDI convergence*. Delft, The Netherlands: Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission, 35–51.

Goodchild, M.F., 2007a. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69, 211–221.

Goodchild, M.F., 2007b. Citizens as voluntary sensors: spatial data infrastructure in the world of Web 2.0. *International Journal of Spatial Data Infrastructures Research*, 2, 24–32.

Goodchild, M.F., Fu, P., and Rich, P., 2007. Sharing geographic information: an assessment of the geospatial one-stop. *Annals of the Association of American Geographers*, 97 (2), 250–266.

Goodwin, J., Hart, G., and Dolbear, C., 2008. Geographical linked data: the administrative geography of Great Britain on the semantic web. *Transactions in GIS*, 12 (Suppl. 1), 19–30.

Gouveia, C. and Fonseca, A., 2008. New approaches to environmental monitoring: the use of ICT to explore volunteered geographic information. *GeoJournal*, 72, 185–197.

Groot, R. and McLaughlin, J., 2000, *Geospatial data infrastructure, concepts, cases and good practice*. Oxford: Oxford University Press, 286p.

Han, H., et al., 2003. Automatic document metadata extraction using support vector machines. In: *Proceedings of the 3rd ACM/IEEE-CS joint conference on digital libraries*, June 2003, Houston, Texas, 37–48.

ISO/TC 211, 2003. *ISO 19115:2003 Geographic Information – metadata*.

ISO/TC 211, 2003. *ISO 19115-2:2009 Geographic information – metadata – Part 2: extensions for imagery and gridded data*.

ISO/TC 211, 2005. *ISO 19110:2005 Geographic information – methodology for feature cataloguing*.

ISO/TC 211, 2005. *ISO 19117:2005 Geographic information – portrayal*.

ISO/TC 211, 2005. *ISO 19119:2005 Geographic information – services*.

ISO/TC 211, 2007. *ISO 19139:2007 Geographic information – metadata – XML schema implementation*.

Kalantari, M., Olfat, H., and Rajabifard, A., 2010, Automatic spatial metadata enrichment: reducing metadata creation burden through spatial Folksonomies. *GSDI 12 World Conference – Singapore* [online]. Available from: http://www.gsdi.org/gsdiconf/gsdi12/papers/23.pdf [Accessed 20 March 2010].

Keller, S.F., 1999. Modeling and sharing geographic data with INTERLIS. *Computers and Geosciences*, 25, 49–59.

Kok, B. and Loenen, B., 2005. How to assess the success of national spatial data infrastructures? *Computers, Environment and Urban Systems*, 29, 699–717.

Kolodziej, K., 2003, *OpenGIS® web map server cookbook* [online]. Available from: [Accessed 20 March 2010]. http://portal.opengeospatial.org/files/?artifact_id=7769

Maguire, D.J. and Longley, P.A., 2005. The emergence of geoportals and their role in spatial data infrastructures. *Computers, Environment and Urban Systems*, 29, 3–14.

Manigas, L., et al., 2009. Implementation of recent metadata directives and guidelines in public administration: the experience of Sardinia Region. *In*: B. Loenen, J.W.J. Besemer, and J.A. Zevenbergen, eds. *SDI convergence*. Delft, The Netherlands: Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission, 151–160.

Manso-Callejo, M.A. and Wachowicz, M., 2009. GIS design: a review of current issues in interoperability. *Geography Compass*, 3 (3), 1105–1124.

Manso-Callejo, M.A., Wachowicz, M., and Bernabé-Poveda, M., 2009, Automatic metadata creation for supporting interoperability levels of spatial data infrastructures. *In*: *GSDI 11 World Conference* [online], 15–19 June 2009, Rotterdam, The Netherlands. Available from: http://www.gsdi.org/gsdiconf/gsdi11 [Accessed 20 March 2010].

Mansourian, A., et al., 2006. Using SDI and web-based system to facilitate disaster management. *Computers and Geosciences*, 32, 303–315.

Masó, J., Pomakis, K., and Julià, N., 2010, *OGC Web Map Tile Service (WMTS) implementation standard, version 1.0.0* [online], OGC 07-057r7. Available from: http://portal.opengeospatial.org/files/?artifact_id=35326. [Accessed 7 January 2011].

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

Masser, I., 2009. Changing notions of a spatial data infrastructure. *In*: B. Loenen, J.W.J. Besemer and J.A. Zevenbergen, eds. *SDI convergence*. Delft, The Netherlands: Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission, 219–228.

Masser, I., Rajabifard, A., and Williamson, I., 2008. Spatially enabling governments through SDI implementation. *International Journal of Geographical Information Science*, 22 (1), 5–20.

Mazzetti, P., Nativi, S., and Caron, J., 2009. RESTful implementation of geospatial services for Earth and space science applications. *International Journal of Digital Earth*, 2 (1), 40–61.

Metadata Ad Hoc Working Group, Federal Geographic Data Committee, 1998. *Content Standard for Digital Geospatial Metadata (CSDGM)*. HTML Version: FGDC-STD-001-1998. Washington, DC: Federal Geographic Data Committee.

Muller, M., 2006. *Symbology encoding (SE) implementation specification, version 1.1.0* [online], OGC 05-077r4. Available from: http://portal.opengeospatial.org/files/?artifact_id=16700 [Accessed 20 March 2010].

Muller, M. and MacGill, J., 2005. *Styled layer descriptor profile of the web map service implementation specification, version 1.1.0* [online], OGC 05-078. Available from: http://portal.opengeospatial.org/files/?artifact_id=22364 [Accessed 20 March 2010].

Na, A. and Priest, M., 2007. *OGC sensor observation service SOS, version 1.0.0* [online], OGC 06-009r6. Available from: http://portal.opengeospatial.org/files/?artifact_id=26667 [Accessed 20 March 2010].

Nebert, D., Whiteside, A., and Vretanos, P.P., 2007. *OGC catalogue service implementation specification, version 2.0.2* [online], OGC 07-006r1. Available from: http://portal.opengeospatial.org/files/?artifact_id=20555 [Accessed 20 March 2010].

Nebert, D.D., 2004. *The SDI cookbook* [online]. Available from: http://www.gsdi.org/docs2004/Cookbook/cookbookV2.0.pdf [Accessed 20 March 2010].

Nedovic-Budic, Z., et al., 2004. Are SDIs serving the needs of local planning. Case study of Victoria, Australia and Illinois, USA. *Computers, Environment and Urban Systems*, 28, 329–351.

Nedović-Budić, Z., Pinto, J.K., and Budhathoki, N.J., 2008. SDI effectiveness from the user perspective. *In*: J.Crompvoets et al., eds. *A multi-view framework to assess SDIs*. Melbourne: The Melbourne University Press, 273–303.

Olfat, H., Rajabifard, A., and Kalantari, M., 2010. Automatic spatial metadata update: a new approach. *In*: *XXIV FIG International Congress – Sidney Australia* [online]. Available from: http://www.fig.net/pub/fig2010/papers/ts05b/ts05b_olfat_rajabifard_et_al_4079.pdf

Olivet, M., et al., 2008. Health services provision and geographic accessibility. *Medicina Clinica*, 131 (4), 16–22.

Ostlander, N., Tegtmeyer, S., and Foerster, T., 2005. Developing an SDI for time-variant and multilingual information dissemination and data distribution. *In*: *11th EC-GI and GIS workshop*. Sardinia, Italy.

Pascual, V., et al., 2009, CatalogConnector. An OGC CSW client to connect metadata catalogues. *GSDI 11 World Conference – Rotterdam, The Netherlands* [online]. Available from: http://www.gsdi.org/gsdiconf/gsdi11/ [Accessed 20 March 2010].

Paudyal, D.R., McDougall, K., and Apan, A., 2009. Building SDI bridges for catchment management. *In*: B. Loenen, J.W.J. Besemer, and J.A. Zevenbergen, eds. *SDI convergence*. Delft, The Netherlands: Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission, 265–279.

Pons, X., 2004. *Manual of MiraMon. Geographic information system and remote sensing software*. Bellaterra: Centre de Recerca Ecològica i Aplicacions Forestals (CREAF) ISBN: 84-931323-5-7.

Rajabifard, A., et al., 2006. The role of sub-national government and the private sector in future spatial data infrastructures. *International Journal of Geographical Information Science*, 20 (7), 727–741.

Rajabifard, A., Feeney, M.E.F., and Williamson, I.P., 2002a. Future directions for SDI development. *International Journal of Applied Earth Observation Geoinformation*, 4 (1), 11–22.

Rajabifard, A., Feeney, M.E.F., and Williamson, I.P., 2002b. The cultural aspects of sharing and dynamic partnerships within an SDI hierarchy. *Cartography Journal*, 31 (1), 1–15.

Rajabifard, A., Kalantari, M., and Binns, A., 2009. SDI and metadata entry and updating tools. *In*: B. Loenen, J.W.J. Besemer, and J.A. Zevenbergen, eds. *SDI convergence*. Delft, The Netherlands: Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission, 121–135.

Scholten, M., Klamma, R., and Kiehle, C., 2006. Evaluating performance in spatial data infrastructures for geoprocessing. *IEEE Internet Computing*, 10 (5), 34–41.

Schut, P., 2007, *OGC web processing service (WPS), version 1.0.0* [online], OGC 05-007r7. Available from:http://portal.opengeospatial.org/files/?artifact_id=24151 [Accessed 20 March 2010].

Schut, P., 2010, *OGC table joining service (TJS) standard, version 1.0.0* [online], OGC 10-070r2. Available from: http://portal.opengeospatial.org/files/?artifact_id=40095 [Accessed 7 January 2010].

Sonnet, J., 2005, *Web map context (WMC) documents, version 1.1.0* [online], OGC 05-005, Available from: http://portal.opengeospatial.org/files/?artifact_id=8618 [Accessed 20 March 2010].

Tait, M.G., 2005. Implementing geoportals: applications of distributed GIS. *Computers, Environment and Urban Systems*, 29, 33–47.

Thompson, J., et al., 2009. SDI for collaborative health services planning. *GSDI11 World Conference–Rotterdam, The Netherlands* [online]. Available from: http://www.gsdi.org/gsdiconf/gsdi11/ [Accessed 20 March 2010].

Tu, S., et al., 2004. Design strategies to improve performance of GIS Web services. *In*: *Proceedings of the International Conference on Information Technology: Coding and Computing*, 444–450.

Vandenbroucke, D., 2008, *Spatial data infrastructures in Europe: state of play 2007*. Belgium: K.U. Leuven (SADL + ICRI).

Vandenbroucke, D., et al., 2009. A network perspective on spatial data infrastructures: application to the sub-national SDI of Flanders (Belgium). *Transactions in GIS*, 13 (s1), 105–122.

Vanderhaegen, M. and Muro, E., 2005. Contribution of a European spatial data infrastructure to the effectiveness of EIA and SEA studies. *Environmental Impact Assessment Review*, 25, 123–142.

van Loenen, B., 2009. Developing geographic information infrastructures: the role of access policies. *Journal of Geographical Information Science*, 23 (2), 195–212.

Vanmeulebrouk, B., et al. 2009. OGC standards in daily practice: gaps and difficulties found in their use. *In*: *GSDI 11 World Conference–Rotterdam, The Netherlands* [online]. Available from: http://www.gsdi.org/gsdiconf/gsdi11/ [Accessed 20 March 2010].

Vretanos, P.A., 2005. *OGC Web feature service (WFS) implementation specification, version 1.1.0* [online], OGC 04-094. Available from: http://portal.opengeospatial.org/files/?artifact_id=8339 [Accessed 20 March 2010].

Wang, S. and Liu, Y., 2009. TeraGrid GIScience gateway: bridging cyberinfrastructure and GIScience. *International Journal of Geographical Information Science*, 23 (2), 169–193.

Whiteside, A. and Evans, J.D., 2008. *OGC web coverage service (WCS) implementation standard, version 1.1.2* [online], OGC 07-067r5. Available from: http://portal.opengeospatial.org/files/?artifact_id=27297 [Accessed 20 March 2010].

Whiteside, A. and Greenwood, J., 2010. *OGC web services common standard (OWS common) version 2.0.0* [online], OGC 06-121r9. Available from: http://portal.opengeospatial.org/files/?artifact_id=38867 [Accessed 8 January 2011].

Wright, D.J., et al., 2003. Why Web GIS may not be enough: a case study with the virtual research vessel. *Marine Geodesy*, 26 (1), 73–86.

Wu, H., et al., 2011. Monitoring and evaluating the quality of web map service resources for optimizing map composition over the internet to support decision making. *Computers and Geosciences*, 37 (1), 485–494.

Wytzisk, A. and Sliwinski, A., 2004. Quo Vadis SDI. *In*: *7 AGILE Conference*, 29 April 29th–1 May, Heraklion, Greece.

Xue, Y., Cracknell, A.P., and Guo, H.D., 2002. Telegeoprocessing: the integration of remote sensing, geographic information system (GIS), global positioning system (GPS) and telecommunication. *International Journal of Remote Sensing*, 23 (9), 1851–1893.

Yang, C., et al., 2005. Performance-improving techniques in Web-based GIS. *International Journal of Geographical Information Systems*, 19 (3), 319–342.

Yang, C., et al., 2006. Spatial Web portal for building spatial data infrastructure. *International Journal of Geographic Information Sciences*, 12 (1), 38–43.

Yang, P., et al., 2007. The emerging concepts and applications of the spatial web portal. *Photogrammetric Engineering and Remote Sensing*, 73 (6), 691–698.

Yang, C. and Raskin, R., 2009. Introduction to distributed geographic information processing research. *Journal of Geographical Information Science*, 23 (5), 553–560.

Zabala, A. and Masó, J., 2005. Integrated hierarchical metadata proposal: series, layer, entities and attributes. *In*: *Proceedings of International Cartographic Conference*, 9–16 July 2005, A Coruña, Spain.

Zabala, A. and Pons, X., 2002. Image metadata: compiled proposal and implementation. *In*: T. Benes, ed. *Geoinformation for all*. Rotterdam, The Netherlands: Millpress, 674–652.

Zimmermann, R., et al., 2006. A distributed geotechnical information management and exchange architecture. *IEEE Internet Computing*, 10 (5), 26–33.

3. BUILDING THE WORLD WIDE HYPERMAP (WWH) WITH A RESTFUL ARCHITECTURE

Aquest capítol és una reproducció de: Masó J., Pons X. i Zabala A. (2012) Building the World Wide Hypermap (WWH) with a RESTful architecture. *International Journal of Digital Earth*. DOI:10.1080/17538947.2012.669414 (en aquest document es referencia com Masó *et al.*, 2012b)

International Journal of Digital Earth,
2012, 1–19, iFirst article

Building the World Wide Hypermap (WWH) with a RESTful architecture

Joan Masó[a]*, Xavier Pons[b] and Alaitz Zabala[b]

[a]*Center for Ecological Research and Forestry Applications (CREAF), Universitat Autònoma de Barcelona, Barcelona, Spain;* [b]*Department of Geography, Universitat Autònoma de Barcelona, Barcelona, Spain*

(*Received 14 September 2011; final version received 21 February 2012*)

The hypermap concept was introduced in 1992 as a way to hyperlink geospatial features to text, multimedia or other geospatial features. Since then, the concept has been used in several applications, although it has been found to have some limitations. On the other hand, Spatial Data Infrastructures (SDIs) adopt diverse and heterogeneous service oriented architectures (SOAs). They are developed by different standard bodies and are generally disconnected from mass market web solutions. This work expands the hypermap concept to overcome its limitations and harmonise it with geospatial resource oriented architecture (ROA), connecting it to the semantic web and generalising it to the World Wide Hypermap (WWH) as a tool for building a single 'Digital Earth'. Global identifiers, dynamic links, link purposes and resource management capabilities are introduced as a solution that orchestrates data, metadata and data access services in a homogeneous way. This is achieved by providing a set of rules using the current Internet paradigm formalised in the REpresentational State Transfer (REST) architecture and combining it with existing Open Geospatial Consortium (OGC) and International Organization for Standardization (ISO) standards. A reference implementation is also presented and the strategies needed to implement the WWH, which mainly consist in a set of additions to current Geographic Information System (GIS) products and a RESTful server that mediates between the Internet and the local GIS applications.

Keywords: SDI; URI; standards; hypermap; RESTful

1. Introduction

Hyperstructure refers to a form of communication that goes beyond or over the linear style so that the reader is able to jump from one content to another through links in a non-linear way (Laurini and Thompson 1992). In fact, in 1967, Douglas C. Engelbart filled a US patent form on a 'x-y position indicator' that was later known as computer mouse (Engelbart 1967) and on 9 December 1968, he publicly demonstrated that a computer mouse could be used to control a networked computer system using hypertext linking among other precursor technologies we use today, from personal computing to social networking (Engelbart and English 1968). The hypertext concept was extended to digital maps and the hypermap concept was established in 1990 by Laurini and Milleret-Raffort (1990) just before

*Corresponding author. Email: Joan.Maso@uab.cat

ISSN 1753-8947 print/ISSN 1753-8955 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/17538947.2012.669414
http://www.tandfonline.com

Berners-Lee proposed the World Wide Web and the hypertext markup language (HTML). Hypermaps are geo-referenced multimedia systems that can hyperstructure individual components with respect to each other and to the map (Kraak and Driel 1997); thus, features on a map can be linked to information of any kind (text documents, pictures, sounds, videos and other multimedia content) through hyperlinks. These links are represented on the screen by buttons appearing as icons/pictograms or typographical enrichments in text, etc. (Boursier and Mainguenaud 1992).

The hyperrelated elements in a hypermap are called resources, and the connections are possible because each resource has an identifier (Laurini and Thompson 1992). The hypermap concept has been implemented in some Geographic Information System (GIS) applications, such as ArcView 3.0 and MiraMon, which evolved from linking to a proprietary data hierarchical structure (Kraak and Driel 1997) to a more flexible link to any document in the computer or over the Internet, including other hypermaps (Pons 2002).

Hypermaps have been used with several purposes: raster images resulting from finite element simulations on urban networks (in which pixels represent identifiers to videos or hypertext) (Saugy *et al.* 1995), in the atlas metaphor (Kraak and Driel 1997), in urban geography and planning learning environments (Thompson *et al.* 1997), in geological studies (Voisard 1998), in intelligent transport systems with geo-referenced video frames (Kim *et al.* 2000) and in tourist maps (Proll 2002). Since the hypermap is closer to a naive user's way of thinking, most of these applications have been developed for the general public in interactive terminals (Boursier and Mainguenaud 1992).

The largest **limitation** of the hypermap is that all queries must be **previously defined** and 'prepared', i.e. links must be established between entities and users can only follow those particular paths (Boursier and Mainguenaud 1992). In addition, resources are linked to other resources by pairing **local identifiers**, and thus limit the scope of their relationships (Yuan *et al.* 2000). Another criticism is that the underlying data **model is extremely poor**, as it only includes the concept of hyperlinks among entities (Boursier and Mainguenaud 1992, Voisard 1998) and has no semantics that identifies the purpose of each hyperlink. Moreover, hypermaps are **limited to retrieving** resources (Voisard 1998) and are not able to create, update or delete resources.

Indeed, Al Gore (1998), in the famous speech about the 'Digital Earth' vision, recognises the need for combining simple users way of thinking to navigate and search geospatial information with a digital globe database maintained by thousands of different producer organisations interconnected by the web, interlinking the information about the planet in a single system. In the past 20 years, new technologies have emerged that are used here to overcome the hypermap limitations and to extend it to a distributed environment for Internet GIS (Yuan *et al.* 2000). In fact, hyperlinks are recognised as a needed functionality for data store interconnection and as user interface navigation elements to build the 'Digital Earth' (Grossner *et al.* 2008). Taking the concept to the extreme, there is only one giant hypermap that links all spatial data in a World Wide Hypermap (WWH). In the present paper, Section 2 covers the technological context that makes the WWH possible, Section 3 describes how these technologies can be applied in the WWH, and Section 4

describes the development of a reference software and some details of its implementation.

2. Technological methodologies

In the introduction, four limitations of the original hypermap concept have been enumerated. This chapter describes four technologies that are useful to extend the hypermap concept, and to evolve it to a WWH and that have been applied to overcome the four limitations identified earlier.

2.1. Geospatial web services and dynamically generated hyperlinks

The original hypermap concept relied on a predefined set of static links; however, this limitation no longer exists thanks to web services. Just sit at your Internet browser and type 'hypermap' in Google search, for instance. You immediately get about 51,000 results (June 2011) in the form of HTML pages with hyperlinks that were dynamically created on the fly by the web service search engine. Similarly, mass market map search engines, like geocommons.com, or catalogues compliant with the Open Geospatial Consortium (OGC) Catalogue Service for the Web (CSW) return collections of dynamically generated links to geospatial data-sets. Indeed, the Internet in general, but particularly the web, has made it possible to share cartographic resources globally. Organisations involved in producing and using cartographical resources have organised themselves into Spatial Data Infrastructures (SDIs), and have eventually created a global community (Coleman and McLaulghlin 1997). SDI implementations are based on web services that make it possible to discover data (web catalogue services), display data (web visualisation services), share data (web downloading services) and even manipulate data (web processing services [WPS]). Classical cartographic products and data-sets are merged with sensor data (remote and *in situ* sensors) and with Volunteered Geographic Information (VGI) (Longueville *et al.* 2010). Distributed services make it possible to build complex systems that rely on smaller pieces or individual corporative services. The European SDI, promoted by the INSPIRE directive (INSPIRE 2008), is an example of a multilevel architecture, and the Global Earth Observations System of Systems (GEOSS) is an example of a more decentralised and diversely composed structure (Butterfield *et al.* 2008).

In fact, current web services implicitly try to extend the hypermap concept. In the SDI, cartographic resources can be immediately interconnected to others. By doing so we are creating a single huge WWH that reuses the hypermap concept but overcomes its main limitations. Current SDIs build an integrated network based on International Organization for Standardization (ISO) and OGC web service (OWS) standards (Percivall 2010), which have helped the development of the SDIs enormously. Web services are distributed applications that manage resources so that the user software no longer accesses the data directly but rather makes requests to remote computer applications. Remote applications obtain a subset of the data and transform it into a suitable format. During this extraction, it is possible to encode hyperlinks to other subsets of the data or to other cartographic resources that were not encoded in the original data; therefore, the hyperlinks are no longer static but rather are constantly created by web services.

Nevertheless, 15 years of using geospatial web services have revealed some practical limitations. One of the main problems is the metadata catalogues and the difficulties involved in generating federated catalogues, resulting in the metadata being replicated in several harvested catalogues that are disconnected from the actual data (Craglia *et al.* 2007). A common SDI has at least two catalogues: one for metadata about data and another for metadata about services. If a user needs some data and makes a query in an SDI search engine, he/she obtains a set of metadata about data that is very often poorly connected to the data and services (data and services are maintained and stored by the data provider itself); or the user receives metadata about data services that often do not have enough information about which data they contain. So even if the user is able to discover the existence of an interesting data-set, he/she will have problems getting it as a file or as a service response. Instead of building a WWH in which the focus is centred on resources, SDIs build infrastructures based on catalogues, which inadvertently confuse users. Furthermore, many current web service standards do not have the transactional extensions to allow them to do more than just access data, but also create and update data. Even if some data standards, such as Geography Markup Language (GML), have the ability to hyperlink geospatial objects, current web services provide poor direct access to hyper-references and do not provide operations for creating new ones. There is also no uniform interface for addressing resources; instead, current standards provide different protocols for the address access to a diversity of resource types, making data exchanges more challenging.

This paper provides a different perspective in which web services are almost invisible in favour of providing immediate access to resources and their associations. Service catalogues are no longer needed for data discovery and access, and only play a role in providing distributed data processing and metadata discovery.

2.2. Global geo-identifiers

Only resources that have a unique identifier can be referenced and indicated by hyper-references in a hypermap. Classic hypermaps use local identifiers to relate internal resources. Replacing local identifiers with global identifiers is the logical solution, but the problem is that global identifiers need a resolver application to find the actual resource that each identifier indicates (Bishr 1999). These resolution systems are too dependent on specific and centralised resolver components. The Internet offers an alternative for identifying geo-resources directly, i.e. Hypertext Transfer Protocol (HTTP) Uniform Resource Identifier (URI), which provide a global addressing space for resource and service discovery. The advantage of HTTP URIs is that they encapsulate all the information required to identify and locate a resource in a global addressing space without needing a centralised registry. This is a simple solution that relies on technological components that have existed for many years, and indeed, make the Web work.

Hypertext Transfer Protocol Uniform Resource Identifiers have some important advantages: They can be used with the current Internet infrastructure and they do not require an extra catalogue component to be maintained. URI can be bookmarked, exchanged via hyperlinks and, given their readability, even advertised (Pautasso *et al.* 2008). These identifiers can be used directly on a REpresentational State Transfer (REST) architecture in REST oriented services (zur Muehlen *et al.*

2005). However, there are some disadvantages (Bishr 1999): Since the system relies on the Internet Protocol (the context in which HTTP URIs are valid), URI cannot work directly on other networks. In addition, it has been argued that the use of URI is too sensitive to the specific location of the server, which can change over time. This problem can be overcome by using Domain Name Servers (DNS) to translate addresses, and thus it is only necessary to configure the networking infrastructure properly (Pautasso *et al.* 2008).

2.3. Hyperlinks with purpose

A classic hypermap link can serve different purposes, but there is no way to express and store these purposes, which is one of the reasons why some people consider the hypermap model to be extremely poor. In the following we give some examples of different purposes described in scientific papers:

- To link resources that are semantically related (Laurini and Milleret-Raffort 1990).
- To link resources to the time when they were created in a way that allows version dependencies to be defined (Voisard 1998, Sargent 1999).
- To link resources with the evidence describing the phenomenon, such as previous documentation, scientific papers, etc. (Voisard 1998).
- To link resources that exist only if other objects exist, such as topological dependencies (Voisard 1998).
- To link multiple representations of the same object in different data-sets or at different scales (Friis-Christensen and Jensen 2005).
- To allow incremental discovery of resources (Pautasso *et al.* 2008).
- To assign a coordinate reference system (CRS) and to define CRS dependencies with a datum ellipsoid, etc., like the GML does (Portele 2007).

Recent web technologies provide different ways of defining the purpose of a hyperlink:

- In an XML document, the xlink arcrole attribute can store the role or purpose of the target resource in relation to the present resource, given as a URI.
- In an HTML page, the Resource Description Framework in attributes (RDFa) allows *rel* and *rev* attributes to be added to express the meaning of a direct or reverse relationship between two resources.

In fact, to express purposes and to achieve the vision of a 'Digital Earth' (Craglia *et al.* 2008) it is necessary to have a precise semantic definition of the concepts used to describe features extracted from the real world. Therefore, the hypermap concept can be used for making connections to the semantic web and ontologies and vocabularies (Bizer *et al.* 2009).

2.4. RESTful web services

Classical hypermaps are limited to retrieving resources. Retrieving resources is only a subset of what Internet architecture can do as it was formalised in the REST

(Fielding 2000). Fielding defines a set of REST principles that can be applied to more generic Resource Oriented Architecture (ROA), a flourishing kind of web services, and particularly geo-services (Richardson and Ruby 2007). REST is an architectural style for building large-scale distributed hypermedia systems (Pautasso *et al.* 2008). It is the perfect environment in which the WWH can grow and break the individual data-set barriers.

The REST architectural style is defined by the following rules: client-server interaction, stateless interaction, cache support, chaining of services, code on demand and uniform interface (Mazzetti *et al.* 2009). In the REST approach, a client performs a request and a server provides a response (client-server interaction) in the form of a document of a particular media type that can sometimes include new code that can be executed on the client (code on demand). No session is established (stateless interaction) and only the information in the request message is used to provide the response (no previous requests from the same client are considered); therefore, the responses can be cached by intermediate elements of the protocol and reused anytime (cache support). Servers can act as clients of other servers, which allows service chaining (chaining of services).

In REST, the client and server interaction is based on exchanging resource representations. Communication of resource representations is possible because each resource has its own resource identifier that can be used in a request (what is known as the addressability principle). A resource representation is a document of some sort (expressed in a particular media type), and some documents contain hyperlinks to other resource identifiers that can be followed to manipulate other resources so that Hypermedia is The Engine Of the Application State (property sometimes referred to with the acronym of HATEOAS). This last property is the same paradigm that Laurini and Milleret-Raffort (1990) used to define the hypermap two decades ago, where a user click on a map feature brings up information and it is a link that can be followed to discover new information that was not directly available. The term RESTful was introduced to indicate 'REST oriented approach' (Richardson and Ruby 2007).

Combining different RESTful requests we can manipulate interrelated resources using resource identifiers without needing previous predefined links. We can also create resources, retrieve representations and update or delete resources, and thus overcome the historical limitations of the classical hypermap.

In this context, the most important characteristic of the REST architectural style is probably the uniform interface. The only mandatory requirement for a RESTful web service is to adopt URI addressing. However, the HTTP application-level protocol is very common. In this case, the HTTP 1.0 methods (sometimes called verbs) (Berners-Lee and Fielding 1996) define the action to be performed on the target resource (GET = retrieve; POST = create; PUT = update; DELETE = delete) (Mazzetti *et al.* 2009). In addition, HTTP 1.0 defines HEAD, and HTTP 1.1 (Fielding *et al.* 1999) adds three additional methods, TRACE, OPTIONS and CONNECT, that can be associated with new actions of the target resource (Pautasso *et al.* 2008). For instance, HEAD can be used to obtain metadata about the resource without retrieving the complete representation of the resource itself, and OPTIONS lists the methods supported by a particular resource (Richardson and Ruby 2007). GET is the most commonly used method on the Web. In fact, we use it many times a day to retrieve web pages on a web browser or to download a PDF file. The POST

operation is used sometimes when we fill in a form on a web page and submit it. Although PUT and DELETE are not used in common web browsing, they are well defined and ready to use for the indicated purposes.

Currently, SDI web services are based on a different architectural style that is called Remote Procedure Call–Service Oriented Architecture (RPC–SOA) (Booth 2004, Whiteside 2005). However, the web 2.0 geo-community is currently pushing the geospatial standard organisations to introduce RESTful styles because they find them simpler to implement. Some RESTful standards based on RDF Site Summary (RSS) and Atom have been extremely successful in the VGI community.

The OCG has recently taken some measures to bring web services closer to the REST style:

- It was recognised that some mass market applications use RESTful services, such as GeoRSS and Keyhole Markup Language (KML) (Percivall 2008).
- A draft for the Integration of ROA Concepts into the OGC Reference Model was proposed (Usländer 2008).
- A RESTful interface for the Workflows and Sensor Web Enablement groups of standards was considered in interoperability experiments, such as the OGC OWS-5 initiative (Werling 2008), tested in the OWS-6 initiative (Cappelaere 2009).
- Open Geospatial Consortium policies mandate that any new OGC standard or new revision of a previous standard must evaluate a ROA style.
- The Web Map Tile Service (WMTS) is the first OGC approved standard that includes an explicitly RESTful interface (Masó *et al.* 2009) (tested in the OGC OWS-6 initiative).

INSPIRE has edited a report that assesses the impact and advantages of ROA and REST (Lucchi *et al.* 2008). This report defines a reference model similar to the one proposed by Usländer (2008) for OGC and provides an extra description of the metadata catalogue use case. It concludes that metadata records in a catalogue can be considered resources and that every metadata record can be accessed by using a URI, without needing the protocol defined in CSW (a query to the catalogue will return a document with hyperlinks to the corresponding URI metadata records that match the request).

3. How to implement the WWH

The four technologies presented earlier are not new but the contribution of this paper is to apply them together in a single system to overcome the four main drawbacks of the old hypermap concept, and make it possible to extend the hypermap so it becomes a WWH. This section describes the essential elements needed to implement them.

If you imagine a permanent collection of data-sets (with all their different components) that represents an immutable world that considers all aspects imaginable, it is easy to generate a set of global identifiers in a centralised way, and then encode the logical links between them. But in the real world, data-sets are generated progressively and frequently updated. A data-set is updated by creating new features or modifying some of the previous ones. However, applications in GIS

analysis currently generate new derived data-sets. This process can be carried out in local GIS applications but also in distributed environments either by professional or VGI communities.

In a ROA the first step is to identify the resources available in the problem and provide a logic that guarantees that each resource has its own global identifier. Resources that are part of the WWH are information such as feature collections (e.g. vector cartographic series), feature instances, feature attributes (textual, multimedia, etc.), coverage collections (e.g. multiband rasters), simple coverages, etc. Previous efforts have been made to apply a ROA style to some of these resources (Mazzetti 2009); however, it is still necessary for a unified protocol to manage all these resources and a mechanism is required to generate and maintain a harmonised but decentralised global geospatial identifiers schema.

The solution presented in this paper proposes combining a new RESTful web service with slightly modified classical GIS applications to generate a seamless WWH environment in which common GIS processes can be carried out directly in a classical GIS application or in a distributed environment on the Web. Figure 1 describes how local GIS instances can access data directly, while external users of the Internet can access data by requesting it from a RESTful application. The following paragraphs explain the details of RESTful access to data and the implications of this approach.

First, global identifiers (HTTP URI) have to be generated as soon as the resource is created. They also have to be maintained during the entire life cycle of these resources, even if elements are generated when the local GIS computer is off-line. This guarantees the uniqueness of the global identifiers and avoids missing links to resources. To make this possible in a decentralised environment, the HTTP URI structure is separated into levels (or namespaces). The first level of the HTTP URI identifies the organisation and is formed by the current URI of its main website followed by the fixed word 'WWH' (using URI template language (Gregorio 2008) this is expressed as http://{OrganizationServer}/WWH). The next level is the name assigned to the software instance used to generate the data (expressed as http://{OrganizationServer}/WWH/{SoftwareInstanceName}). The third level is the name of a geospatial set of entities (e.g. a feature or a coverage collection) (expressed as: http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}. The next levels can identify particular features, coverages or even feature attributes of the geospatial entities and so on as explained in Table 1.

The uniqueness of the complete URI depends on three independent pieces of software that manage the first, second and further levels defined earlier. The uniqueness of the first level is provided by the HTTP naming authorities and

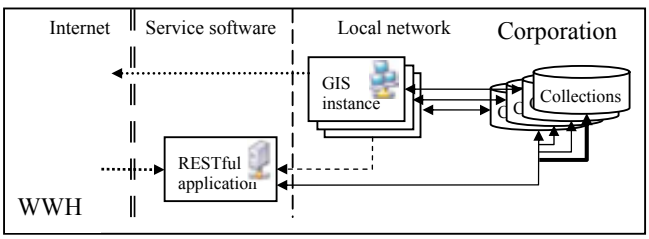


Figure 1. WWH corporative node elements.

Table 1. Basic geospatial URI templates in the WWH.

Resource	URI template
Organisation	http://{OrganizationServer}/WWH
Software instance	http://{OrganizationServer}/WWH/{SoftwareInstance}
Geospatial collection (vector features, raster coverages or record sets)	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}
Geospatial entity	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}
Feature attribute	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}/{attribute}
Coverage element cell value	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}/{row}/{col}
Record set cell value	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}/{RecordSet}/{record}/{field}

mechanisms (mainly the DNS registration). The uniqueness of the second level is provided by a centralised RESTful application inside the corporative domain. The uniqueness of the following levels is provided by each GIS software instance.

Each GIS software instance in the corporation can manage their own resources. Creating a new geospatial collection resource implies creating a new identifier in the GIS software instance namespace. Next, an internal identifier (unique in the geospatial collection namespace) is associated with each feature created in the geospatial collection (several GIS software already do this).

Geospatial collections can be composed of internal entities that have identifiers in the geospatial collection namespace, or by external geospatial entities that have external geospatial identifiers in other namespaces that can be stored in the same corporation or in any other. Geospatial collections can contain vector, raster and tabular data in an integrated way. For vector data, a geospatial collection is a feature collection that has vector features (geospatial entities) which have attributes. Each attribute is associated with a feature (including the geometries) and receives a unique identifier (See Table 1). This schema can go even deeper to geometry coordinates or types that are composed by complex attributes, but these are more advanced capabilities. For raster data the geospatial collection is formed by several raster layers, each describing a value distribution. In this case the atomic resource is the value of a cell of a single raster layer (coverage element). For tabular data, the geospatial collection is a database that is formed by record sets (tables or predefined queries). The atomic resource is the cell value of a particular field in a particular record.

Once identified, these elements can be hyperlinked to internal or external resources. In the WWH model, a hyperlink is a unidirectional relation between two resources of the WWH. Links use global identifiers, and a purpose for this relation can also be included. Yuan *et al*. (2000) described some examples of purposes in hypermaps, such as 'part and whole', 'generalization', 'more resolution', 'previous version', and 'new version'. These purposes can be combined with other exhaustive lists of types of links, such as the ones described by Kopak (1999), and extended with

concepts needed for the resource management system, such as 'copied from' and 'moved to', as it will be described later.

Following this schema, GIS software instances are able to interact directly with their resources in their usual way (binary file accessing, geodatabase querying, etc.) and are also able to interact with remote features using the RESTful requests to the corresponding corporative application by applying one of the four HTTP methods and a global resource identifier. The four HTTP methods in the REST architecture (POST, GET, PUT and DELETE) are associated with the four common actions of any computer system (create, retrieve, update and delete) for each resource type. To create a resource, you have to send it with a POST operation to the parent resource URI. The response of that request is the new resource URI. To update a resource, the new description of the resource has to be sent to the resource URI in a PUT operation. In a system with versioning, the previous version of the resource could be hyperlinked to the new one. To delete a resource, a DELETE operation has to be sent to the resource URI. Finally, a GET operation with the URI retrieves a description of the resource.

On the web, discovering resources is simple because the URI of any HTTP server has a single entry point. In the WWH, discovering resources in a corporation from a client that is external to it is also simple because there is also only one entry point to the corporative part of the WWH (http://{OrganizationServer}/WWH) that indicates the corporative RESTful application. A GET request to this URI returns a description of the service and also enumerates the public software instances that are present in this corporation. Then a GET operation to the http://{OrganizationServer}/WWH/{SoftwareInstance} enumerates the geospatial collections available in the service. A GET request to http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection} enumerates the URIs of the geospatial entities available. Table 2 lists some examples of possible operations that can be carried out on geospatial entities.

Figure 2 uses Unified Modelling Language (UML) diagrams to provide a summary of resources and their operations, relations and URI templates in the WWH. To create these diagrams, we followed the rules described by Rauf (2010).

Table 2. Retrieval, creation, update and delete of geospatial entities.

HTTP URI template	Method	Action
http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}	GET	Returns the geospatial entity and its attribute values.
http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}	POST	Creates a new entity. It returns the geospatial entity identifier.
http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}	PUT	Replaces the most recent version of the geospatial entity.
http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}	DELETE	Removes a particular geospatial entity.

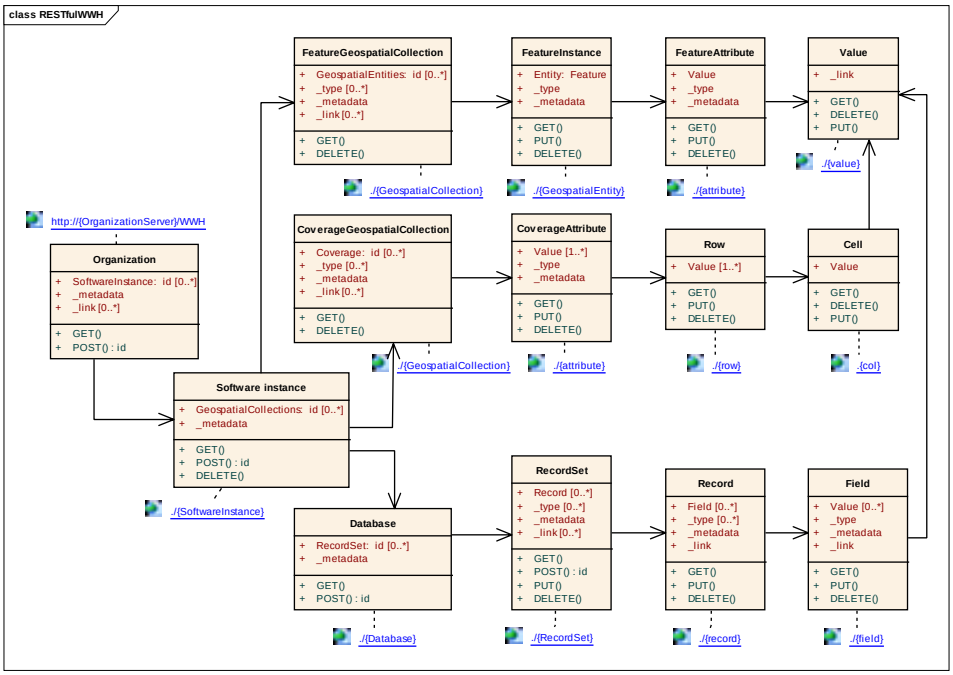


Figure 2. Summary of resources and their operations, relations and URI templates in the WWH.

There are five special resources that can be used by adding the words;_metadata', '_type', '_link', '_all' and '_history' to an appropriate resource. For instance, by adding _metadata to http://{OrganizationServer}/WWH we can refer to the RESTful service metadata in the ISO 19139 style with the list of its capabilities; or by adding it to http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection} we obtain a data-set metadata document, also in the ISO 19139. By adding '_type' to a vector geospatial collection URI we can access a description of the data model of geospatial entities mainly composed by attribute types that can be classified into spatial (geometrical), thematic (textual descriptions of properties, multimedia types, etc), and temporal types. These data models can be transmitted as GML application schemas or in ISO 19110 descriptions encoded in XML. By adding '_type' to a multidimensional data coverage collection URI we obtain the data model describing the dimensions of the raster and the value types of each coverage element. In this case, only the methods PUT and GET (update and retrieve) apply since metadata are automatically associated with the geospatial collection during their creation and it makes no sense to delete them without deleting the collection itself. Adding '_link' to any resource makes it possible to access a list of links for this resource. A link provides a target URI and an optional purpose for the link. We added another reserved word, '_all', to indicate all child resources of a particular resource. This is needed in certain cases for specific purposes, for instance to obtain the data model description of a particular property for all feature instances of a geospatial collection using this URI http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_all/{attribute}/_type.

Sometimes all the resources of a particular level have a common child resource. In this case, we use the reserved word all to obtain a URI that follows a common pattern but affects a group of resources. An example of this practice is when the value array of a particular cell of a coverage collection (a value for each coverage element) needs to be obtained: http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_all/{row}/{col}.

In a real system, some resources have to be updated due to a change in the real world or due to a better interpretation of the same phenomena. A resource identifier of an updated resource will be the same as the previous one. To access an old version of it, the reserved word '_history' can be added to obtain a list of the previous versions and their identifiers that will follow the pattern _history/r1, _history/r2, etc. It will also be possible to obtain the version that was valid on a particular date by adding this URI fragment to the end of any resource: /_history?date = {date}.

Moving or deleting resources is particularly critical in a distributed environment in which external resources might be linked to the ones that are moved or deleted. If a resource needs to be moved, it receives a new identifier in the target resource collection and a hyperlink is set on the source resource. The original identifier cannot be used in the target resource collection because the decentralised method of resolving identifiers is based on the actual hierarchy of the identifier itself. Moving a resource implies retrieving a resource representation in order to do three things: create a new resource in the target resource collection, create the hyperlink in the source with the 'moved' purpose and delete the original resource (keeping the hyperlink active). Therefore, any external link to the original resource is still valid. Deleting resources is not recommended and resource identifiers are never deleted. It is possible to delete the content of a resource but not the identifier itself because it could be the target of hyperlinks.

If a particular resource is copied to another software instance, it keeps its original resource identifier. The new software instance is capable of knowing that this is a copy of an external resource because resource identifiers do not share the same structure as internal objects. If any child resource is modified or created by the software instance, communication is established with the original corporative software to determine whether the original copy has been modified and to request that this resource collection is edited. If communication with it is possible, any modification in the source is updated in both copies. If communications are off-line the geospatial identifier changes to a new one under the software instance namespace. A link is set between both geospatial collections with the 'copied from' purpose. The unedited child resource still has its original identifier. Synchronisation procedures can be executed when communications are possible again.

Apart form retrieving a single object, a very common operation with feature collections is to extract a subset of them according to some spatial and/or thematic criteria. This functionality can be achieved by the RESTful encoding with a POST request by adding the words _SpatialSubset and/or _FilteredSubset to a geospatial collection URI that includes a spatial object description encoded in, for example, GML, or a reference to it (i.e. a direct reference or an OGC Web Feature Service [WFS] request that retrieves an object, typically a rectangular bounding box). The result is a URI of a new geospatial collection that can be retrieved later like any other.

3.1. Catalogues in the WWH

The proposed solution has several advantages over a classical approach based on RPC web services. It eliminates the need for specific data access web services as well as different identifiers or data, metadata and data services to the data. Actually, using a single identifier and a simple set of RESTful rules, users can indicate the data, obtain their metadata and gain access to the actual data. WWH also makes it completely unnecessary to catalogue data access services. OGC CSW metadata catalogues can be used to catalogue and search in metadata about data but it is also expected that common text searching web tools and crawlers will also be useful for discovering data in the WWH.

Most of the metadata catalogues cannot automatically discover new resources by themselves (except if they are harvesting another catalogue), and regular manual updates imply a large investment of time. In essence, the corporative RESTful application acts as an automatic corporative metadata catalogue and presents metadata and data simultaneously to the authorised users. Moreover, in the presented REST architecture, the current web crawlers are able to automatically discover by simply reading the HTTP GET response of http://{OrganizationServer}/WWH (a simple crawler could use the list of DNS names). The crawler can then follow the links to other resources and request descriptions recursively and catalogue resources automatically. Moreover, the existence of the '_link' special resource allows links to other resources to be exposed. Therefore, crawlers can discover which resources are linked more frequently and thus gain some criteria with which to arrange the resources by relevance. This does not invalidate the existence of OGC CSW data catalogues; on the contrary, it provides a way that they can function more automatically and collect all the available information, filtering by any criteria. It also allows popular web search engines to include geospatial information and expand the availability and accessibility of geographic information to the general public.

Furthermore, catalogues can rely on the uniqueness of the original resource URI, so there is no need to generate their own local identifiers. The generation of different identifiers for the same resource is a problem that affects many current catalogues (Nebert 2004). Another advantage is that metadata catalogues use the data URI as the base identifier so the responses of the catalogue will immediately link to the actual data and users will know how to get them immediately.

3.2. Services in the WWH

Having an identifier for each geospatial resource from the moment the resource was created provides automatic access to any geospatial resource, and therefore there is no need for a specific feature or coverage service protocols. As the data access service is implicit in the WWH there is also no need to catalogue data access services anymore. Therefore, this solution replaces current OGC WFS, WFS Transactional (WFS-T) and Web Coverage Service (WCS) protocols and functionalities and makes use of diverse OGC format encodings to exchange information. In fact, encodings are used to communicate resource representations and GML, KML, GeoRSS (Wick and Becker 2007) and Filter Encoding language are excellent recommended encodings for geospatial representations in the WWH. However, map services, including the tiled ones, can be offered by each geospatial collection directly using

the Web Map Service (WMS) (de la Beaujardiere 2004) or WMTS protocols, considering the geospatial collection URI as a root server of the OGC service and adding the WMS Key-Value Pair (KVP) specified parameters (KVP or REST for the WMTS).

Finally, the RESTful application can be extended to provide remote execution support for processes with internal or external data. Processes can be executed directly by the GIS application or can be executed by posting an OGC WPS execute document (Schut 2007) to the http://{OrganizationServer}/WWH/_SpatialOperation URI. The response is a WPS execute document with the result or results, which can be numeric values, texts, tables, or URIs of each geospatial result in the form of a geospatial collection in the http://{OrganizationServer}/WWH/_external namespace. A list of the URIs of the available processes in the server can be retrieved with a GET operation to the http://{OrganizationServer}/WWH/_SpatialOperation URI. The description of the processes can be retrieved in the same format described in the OGC WPS DescribeProcess operation, through a GET operation to http://{OrganizationServer}/WWH/_SpatialOperation/{Process}.

4. Implementation example

The implementation of the WWH requires some extra software that can run on top of the HTTP protocol. As a reference implementation this paper presents how it was implemented in an extended version of the MiraMon software. Here we used a file based implementation, but it is possible to implement it using other data systems, such as a geodatabase.

MiraMon is a GIS and Remote Sensing software that allows visualisation, query, edition and analysis of raster (remote sensing images, orthophotos, digital terrain models, conventional thematic maps with raster structure, etc.) and vector (topographic and thematic maps that contain points, lines or polygons) data-sets, and also connects to OGC services (e.g. WMS and WMTS). Attributes are stored in relational database tables. MiraMon is developed cooperatively by various members of the Consolidated Research Group GRUMETS of the Centre for Ecological Research and Forestry Applications (CREAF) and the Autonomous University of Barcelona (UAB) (Pons 2002).

MiraMon Map Server (MMS) is a component of this software that provides web services in protocols and standards established by the OGC like WMS, WMTS, WCS and WFS. The server application is a Common Gateway Interface (CGI) type executable, developed in C language, which can be directly installed on a web server for Windows (Internet Information Server, Apache, etc.).

The extension of the MiraMon software can be explained as a set of modifications on the MiraMon GIS Professional software and the extension of MMS into a RESTful service. Figure 3 shows how a corporation uses a single instance of the RESTful application and a single entry port that can manage several GIS instances, and how two particular imaginary corporations are interconnected in the WWH. The two applications share a common core of functionalities but also have some particularities. In the common core, both applications are able to perform the maintenance of the internal data files directly and both can act as RESTful clients using HTTP. They are able to make requests like the ones listed in Tables 1 and 2 and explained in Section 3.

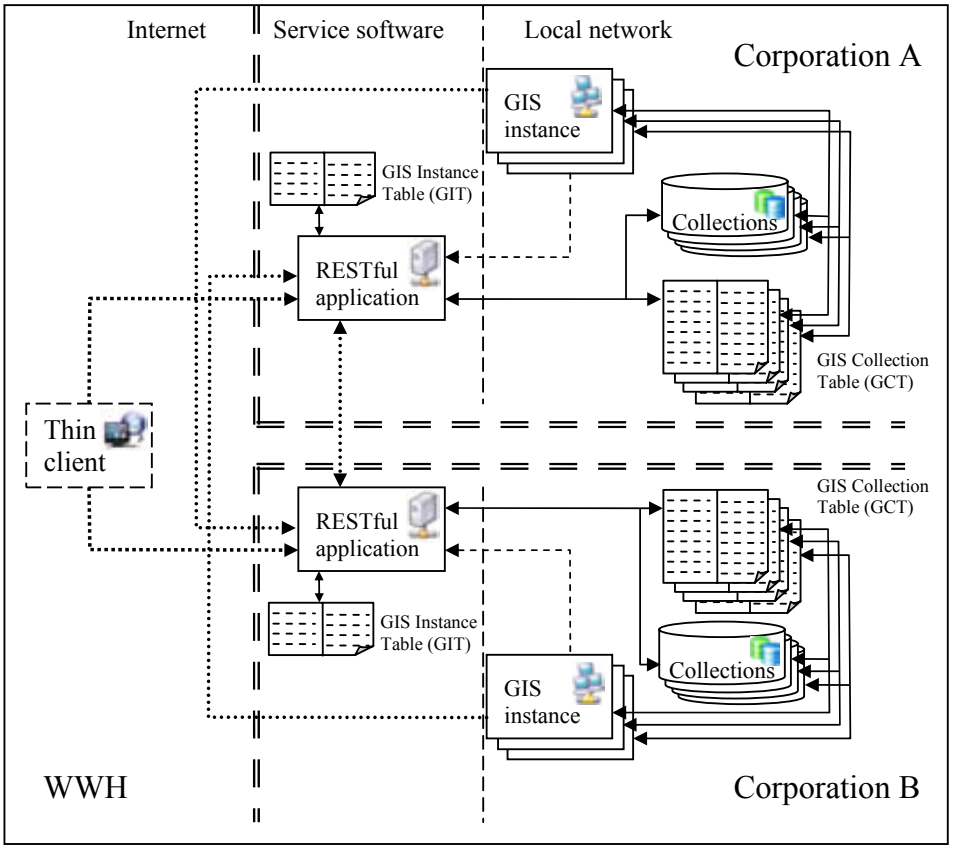


Figure 3. Software components and connections between two corporate nodes and an external thin client.

4.1. Modifications in the GIS internal component

The main functionalities of the MiraMon software have not been altered, but small additions have been made. The internal file data model has been modified (keeping backwards compatibility) to allow the collection identifier (recorded as a metadata field) to be stored, and hyperlinks to geospatial collections (recorded as metadata fields) and to geospatial entities and attributes (stored in the attributes database). Support for references to external geospatial entities in geospatial collections has also been added. The ability to function as a RESTful client (and make a request to any other RESTful service in the WWH) has been added.

The MiraMon software uses the methodology previously described to access the URIs until it gets the list of geospatial collections available. The geospatial collections are then transferred and shown to the user. Another important modification is the addition of the geospatial collection table (GCT) that keeps track of the identifier of all the geospatial collections (data-sets) maintained by a particular GIS instance and the access method required for managing the data (e.g. a file path or the geodatabase access parameters).

4.2. Corporative RESTful server

This application is an HTTP CGI capable of responding to RESTful requests. It recognises the four HTTP methods and translates them into actions that have to be performed on the identified resources. It is able to recognise a global resource identifier, to find the GIS instance that keeps the data, discover where the actual resource is stored and to perform the requested operation directly on the data. To make this possible, it keeps a list of all GIS instances in the corporation (in a GIS instance table, GIT). When a new GIS software instance is set-up on a particular internal computer it also sets up its own GCT. In addition, it creates a resource identifier with a POST operation to the http://{OrganizationServer}/WWH URI sending information about how to access the GCT. This makes the RESTful service register the new GIS instance in the GIT.

When a corporate RESTful application receives a request for internal data it receives the global identifier. This identifier can be separated into three levels. The first level is used to identify the request as a URI to an internal resource. The second level allows it to identify the software instance and read the GIT to determine the way to access the GIS instance GCT table. The third level provides information about the geospatial collection requested; the GCT translates it to information on how to access the real data that can be accessed directly. The RESTful implementation can also act as a GIS software so that it has a reserved software instance identifier, http://{OrganizationServer}/WWH/_external, and also manages its own GCT for collections that are created from outside without any internal GIS software instance intervention, such as requests for spatial filters, thematic filters or process results, or by simply sending new data with a POST operation.

The proposed implementation of the WWH concept demonstrates that a RESTful approach provides a unified interface for discovering data as well as accessing it and executing processing services. Current systems use a diversity of standards to do so and current implementations fail to provide a global unified identification system, so that users can not easily visualise or download the data once discovered in metadata catalogues. Metadata producers are currently required to manually include references to map services and downloading services that are considered optional in ISO metadata model and they are usually left blank. The proposed WWH implementation implicitly and automatically connects all services together. Additionally, the system simplifies the metadata cataloguing process by relying on web crawlers instead of requiring specific geospatial metadata catalogues.

5. Conclusions

This paper demonstrates that the RESTful architecture can reproduce the hypermap concept defined two decades ago and overcome its original limitations. The ROA hides the complexity of the web services and unifies data, metadata and services, reusing simple principles that have been used in the web from the beginning. A network of dynamic hyperlinked geospatial resources is accessed using global URIs in a distributed environment that defines a single huge hypermap.

Current SDIs have implemented RPC–SOA style services based on OGC and ISO international standards for over 10 years, while mass market web services and VGI communities mainly use RESTful services. The WWH concept reconnects these

two worlds again and lowers the entry barrier, providing a harmonised environment, adopting the semantic web and contributing to achieve the 'Digital Earth vision'. GEOSS and INSPIRE are only two examples of large geospatial data systems composed by a distributed collection of data servers and systems that use heterogeneous data access standards and protocols to disseminate scientific and commercial data. By adopting many different standards, both improve interoperability but still face some difficulties. INSPIRE attempts to overcome these difficulties by defining guidelines and implementation rules, narrowing the standard set to a common profile for discovery under a common data access framework. Despite these efforts, none of these systems have an integrated mechanism for upgrading and maintaining new data. Implementations of the WWH proposed in this paper will provide a unified interface for discovering data as well as accessing data and services, and will interrelate knowledge by improving access to geospatial data and the results of scientific studies.

The proposed system combines common metadata catalogues with current search engine tools and makes cataloguing data access services unnecessary, as it provides an implicit way to access data. By adopting a simple set of rules, and based on current standard data formats and encodings, the system provides an interface that can also create and edit geospatial data remotely. Common functionalities, such as spatially or thematically filtering geospatial collections, generating maps, executing processes, etc., are also included in the RESTful approach, making data access services (such as WFS and WCS) unnecessary and integrating other web service standards such as WMS, WMTS and WPS.

Finally, the paper also presents a reference implementation that demonstrates that only minor extensions and modifications to a state-of-the-art GIS software are necessary so that it can be included in the WWH. A corporate RESTful CGI application acts as a broker, connecting local applications and files to the WWH. Future work will aim to implement secured access to selected resources and harmonise them with current sensor web protocols.

Acknowledgements

This paper was written with the support of the European Commission through the FP7-265178-GeoViQua (ENV.2010.4.1.2-2) and a grant to the Consolidated Research Groups given by the Catalan Government (2009 SGR 1511). Xavier Pons is recipient of an ICREA Acadèmia Excellence in Research grant (2011–2015).

References

Berners-Lee, T. and Fielding, R., 1996. *Hypertext Transfer Protocol -- HTTP/1.0*. Available from: http://www.ietf.org/rfc/rfc1945.txt [Accessed 23 June 2011].

Bishr, Y. 1999. A global unique persistent object ID for geospatial information sharing. *In*: A. Vckovski, K.E. Brassel, and H.-J. Schek, eds. *Interoperating geographic information systems.* Berlin: Springer Lecture Notes in Computer Science, No. 1580, 55–64.

Bizer, C., Heath, T., and Berners-Lee, T., 2009. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5 (3), 1–22.

Booth, D., *et al*., 2004. *Web Services Architecture, W3C Working Group Note 11 February 2004*. Available from: http://www.w3.org/TR/ws-arch/wsa.pdf [Accessed 23 June 2011].

Boursier, P. and Mainguenaud, M. 1992. Spatial query languages; extended SQL vs visual languages vs hypermaps. *In: Proceedings of the 5th International Symposium on Spatial Data Handling*, August 1992, Charlestopn, VA, 242–295.

Butterfield, M.L., Pearlman, J.S., and Vickroy, S.C., 2008. A system-of-systems engineering GEOSS; architectural approach. *IEEE Systems Journal*, 2 (3), 321–332.

Cappelaere, P.G. 2009. OWS-6 RESTful Workflow Architecture ER. OGC. Available from: https://portal.opengeospatial.org/files/?artifact_id=33731 [Accessed 9 March 2012].

Coleman, D. and McLaulghlin, J.D., 1997. Information access and network usage in the emerging spatial information marketplace. *Journal of Urban and Regional Information Systems Association*, 9 (1), 8–11.

Craglia, M., *et al*., 2008. Next-generation Digital Earth. *International Journal of Spatial Data Infrastructures Research*, 3, 146–167.

Craglia, M., Kanellopoulos, I., and Smits, P. 2007. Metadata where we are now, and where we should be going. *10th AGILE International Conference on Geographic Information Science*. Aalborg University, Denmark, 1–7. Available from: http://people.plan.aau.dk/~enc/AGILE 2007/PDF/93_PDF.pdf [Accessed 9 March 2012].

de la Beaujardiere, J., 2004. *OGC Web Map Service (WMS) Interface*, Ver 1.3.0, OGC 03-109r1. Available from: http://portal.opengis.org/files/?artifact_id = 5316 [Accessed 20 March 2010].

Engelbart, D.C., 1967, *X-Y Position Indicator for a Display System*. US Patent 3541541. 17 November 1970.

Engelbart, D.C. and English, W.K. 1968, A research center for augmenting human intellect. *AFIPS Conference Proceedings of the 1968 Fall Joint Computer Conference*, 9–11 December 1968, San Francisco, CA, 33, 395–410.

Fielding, R.T., 2000. *Architectural styles and the design of network-based software architectures*. Thesis (PhD). University of California.

Fielding, R.T., *et al*., 1999. *Hypertext Transfer Protocol -- HTTP/1.1*. Available from: http://www.ietf.org/rfc/rfc2616.txt [Accessed 23 June 2010].

Friis-Christensen, A. and Jensen, C.S., 2005. A conceptual schema language for the management of multiple representations of geographic entities. *Transactions in GIS*, 9 (3), 345–380.

Gore, A. 1998. *The digital earth: understanding our planet in the 21st century*. Available from http://www.nextspace.co.nz/wp-content/uploads/2010/08/Gore-Speech2.pdf [Accessed 4 February 2012].

Gregorio, J., 2008. *URI Template*. Network Working Group Internet- Draft. Available from: http://tools.ietf.org/html/draft-gregorio-uritemplate-03 [Accessed 14 September 2011].

Grossner, K.E., Goodchild, M.F., and Clarke, K.C., 2008. Defining a Digital Earth System. *Transactions in GIS*, 12 (1), 145–160.

INSPIRE, 2008. Directive of the European Parliament and of the Council Establishing an Infrastructure for Spatial Information in the Community (INSPIRE).

Kim, Y.I., Pyeon, M.W., and Eo, Y.D., 2000. Development of hypermap database for ITS and GIS. *Computers, Environment and Urban Systems*, 24, 45–60.

Kopak, R.W., 1999. Functional link typing in hypertext. *ACM Computing Surveys*, 31 (4es), 1–5.

Kraak, M.J. and Driel, R.V., 1997. Principles of hypermaps. *Computers and Geosciences*, 23 (4), 451–464.

Laurini, R. and Milleret-Raffort, F., 1990. Principles of geomatic hypermaps. *Proc. 4th Intern. Symposium on Spatial Data handling*. Zurich, 2, 642–655.

Laurini, R. and Thompson, D. 1992. *Fundamentals of spatial information systems*. London: Academic Press, 680, Chapter 16.

Longueville, B. de, *et al*., 2010. Digital Earth's Nervous System for crisis events: real-time Sensor Web Enablement of Volunteered Geographic Information. *International Journal of Digital Earth*, 3 (3), 242–259.

Lucchi, R., Millot, M., and Elfers, C. 2008. Resource Oriented Architecture and REST. Assessment of impact and advantages on INSPIRE. EUR 23397 EN – 2008.

Masó, J., Pomakis, K., and Julià, N. 2009. *OGC Web Map Tile Service (WMTS) Implementation Standard*, Ver 1.0.0, OGC 07-057r7. Available from: http://portal.opengeospatial.org/files/?artifact_id = 35326 [Accessed 7 January 2011].

Mazzetti, P., Nativi, S., and Caron, J., 2009. RESTful implementation of geospatial services for Earth and Space Science applications. *International Journal of Digital Earth*, 2 (1), 40–61.

Nebert, D.D. 2004. *The SDI Cookbook*. Available from: http://www.gsdi.org/docs2004/Cookbook/cookbookV2.0.pdf [Accessed 20 March 2010].

Pautasso, C., Zimmermann, O., and Leymann, F. 2008. RESTful Web Services vs. Big Web Services; Making the Right Architectural Decision. *WWW 2008*, Beijing, China, 21–25 April.

Percivall, G. 2008. *OGC Reference Model*. OGC 08-062r4. Available from: http://portal.opengeospatial.org/files/?artifact_id = 31112 [Accessed 12 December 2009].

Percivall, G., 2010. The application of open standards to enhance the interoperability of geoscience information. *International Journal of Digital Earth*, 3 (S1), 14–30.

Pons, X. 2002. MiraMon. Geographic Information System and Remote Sensing software. *Centre de Recerca Ecològica i Aplicacions Forestals, CREAF*. Bellaterra. ISBN: 84-931323-5-7.

Portele, C. 2007. *OpenGIS® Geography Markup Language (GML) Encoding Standard*. Ver 3.2.1, OGC 07-036. Available from: http://portal.opengeospatial.org/files/?artifact_id=20509 [Accessed 14 September 2011].

Proll, B. 2002. Exploring Hypermedia Design Patterns in Web-based Tourism Information Systems. *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics* (SCI 2002), Orlando, USA, 14–18 July.

Rauf, I., *et al*., 2010. Modeling a Composite RESTful Web Service with UML. *Proceedings of the Fourth European Conference on Software Architecture ECSA*. Companion Volume. University of Copenhagen. 23–26 August.

Richardson, L. and Ruby, S., 2007. *RESTful web services*. Sebastopol, CA: O'Reilly Media.

Sargent, P., 1999. Features identities, descriptors and handles. *In*: A. Vckovski, K.E. Brassel, and H.J. Schek, eds. *Interoperating geographic information systems*. Berlin: Springer Lecture Notes in Computer Science No. 1580, 41–53.

Saugy, B., *et al*., 1995. GigaView and Hypermaps for Finite Elements Based Urban Networks and System Managers. *Computers, Environment and Urban Systems*, 19 (3), 151–160.

Schut, P. 2007, *OGC Web Processing Service (WPS)*, Ver 1.0.0, OGC 05-007r7. Available from: http://portal.opengeospatial.org/files/?artifact_id = 24151 [Accessed 20 March 2010].

Thompson, D., *et al*., 1997. Towards a framework for learning with GIs; The case of UrbanWorld, a hypermap learning environment based on GIS. *Transactions in GIS*, 2 (2), 151–167.

Usländer, T., 2008. Integration of Resource-Oriented Architecture Concepts into the OGC Reference Model. OGC 07-156r1 internal draft document.

Voisard, A. 1998. Geologic hypermaps are more than clickable maps. *Proceedings of the 6th ACM international symposium on Advances in GIS*. Washington, DC, 14–19.

Werling, M. 2008. *OWS-5 GeoProcessing Workflow Architecture Engineering Report*. Ver 0.1.0. OGC 07-138r1. Available from: http://portal.opengeospatial.org/files/?artifact_id = 30065&version = 1 [Accessed 12 December 2009].

Whiteside, A. 2005. *OpenGIS web services architecture description*. OGC 05-042r2 Best Practices Paper.

Wick, M. and Becker, T., 2007. Enhancing RSS feeds with extracted geospatial information for further processing and visualization. *In*: A. Scharl and K. Tochtermann, eds. *The Geospatial Web. How Geobrowsers, Social Software and the Web 2.0 are Shaping the Network Society*. Berlin: Springer, 105–115.

Yuan, X., Chen, N., and Gong, J., 2000. A distributed hypermap model for Internet GIS. *Geo-Spatial Information Science*, 3 (4), 9–15.

zur Muehlen, M., Nickerson, J.V., and Swenson, K.D., 2005. Developing web services choreography standards—the case of REST vs. SOAP. *Decision Support Systems*, 40, 9–29.

4. OPENGIS WEB MAP TILE SERVICE IMPLEMENTATION STANDARD

Aquest capítol és una reproducció de: Masó J., Pomakis K. and Julià N. (2010), Web Map Tile Service Implementation Standard (WMTS). OGC 07-057r7. *Open Geospatial Consortium*. http://www.opengeospatial.org/standards/wmts (en aquest document es referencia com Masó *et al.*, 2010a)

Open Geospatial Consortium Inc.

Date: 2010-04-06

Reference number of this document: OGC 07-057r7

Version: 1.0.0

Category: OpenGIS® Implementation Standard

Editors: Joan Masó, Keith Pomakis and Núria Julià

OpenGIS® Web Map Tile Service Implementation Standard

Copyright © 2010 Open Geospatial Consortium Inc.
To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.

Warning

This document is an OGC Member approved international standard. This document is available on a royalty free, non-discriminatory basis. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation..

Document type: OpenGIS® Standard
Document subtype: OGC Standard
Document stage: Approved for release
Document language: English

Contents

Page

i. Preface ... viii
iii. Submitting organizations ... viii
iv. Document contributor contact points ... viii
v. Revision history ... ix
vi. Changes to the OGC Abstract Specification ... x
vii. Changes to OpenGIS® Implementation Standards ... x
viii. Future work ... x

Foreword ... xi

Introduction ... xii

1 Scope ... 1
2 Compliance ... 1
3 Normative references ... 1
4 Terms and definitions ... 2
5 Conventions ... 3
5.1 Abbreviated terms ... 3
5.2 UML notation ... 4
5.3 Used parts of other documents ... 4
5.4 Platform-neutral and platform-specific standards ... 4
5.5 UML graphical and table representations ... 4
6 WMTS overview ... 6
6.1 Tile matrix set – the geometry of the tiled space ... 7
6.2 Well-known scale sets ... 10
7 WMTS Implementation model ... 11
7.1 Service metadata ... 11
7.1.1 ServiceMetadata document ... 12
7.1.1.1 ServiceMetadata document description ... 12
7.1.1.2 ServiceMetadata document XML schema ... 30
7.1.1.3 ServiceMetadata document example ... 31
7.1.2 GetCapabilities operation (mandatory in procedure oriented architectural style) ... 36
7.1.2.1 GetCapabilities operation request ... 36
7.1.2.2 GetCapabilites exceptions in procedure oriented architectural style ... 38

 Copyright © 2010 Open Geospatial Consortium Inc.

7.1.3 ServiceMetadata resource request (mandatory in resource oriented architectural style)....39
7.2 Tile....39
7.2.1 Tile resource....40
7.2.2 GetTile operation (mandatory in procedure oriented architectural style)....40
7.2.2.1 GetTile operation request....40
7.2.2.2 GetTile exceptions in procedure oriented architectural style....43
7.2.3 Tile resource request (mandatory in resource oriented architectural style)....44
7.3 FeatureInfo....44
7.3.1 FeatureInfo document....44
7.3.2 GetFeatureInfo operation (optional in procedure oriented architectural style)....45
7.3.2.1 GetFeatureInfo operation request....45
7.3.2.2 GetFeatureInfo exceptions in procedure oriented architectural style....47
7.3.3 FeatureInfo resource request (optional in resource oriented architectural style)....49
7.4 Operation request encoding....49
8 WMTS using HTTP KVP encoding....49
8.1 GetCapabilities....50
8.1.1 GetCapabilities request HTTP KVP encoding....50
8.1.2 GetCapabilities request HTTP KVP encoding example....50
8.1.3 GetCapabilities HTTP KVP encoding response....50
8.1.4 GetCapabilities HTTP KVP encoding response example....50
8.2 GetTile....51
8.2.1 GetTile request HTTP KVP encoding....51
8.2.2 GetTile request HTTP KVP encoding example....51
8.2.3 GetTile HTTP KVP encoding response....52
8.2.4 GetTile HTTP KVP encoding response example....52
8.3 GetFeatureInfo....52
8.3.1 GetFeatureInfo request HTTP KVP encoding....52
8.3.2 GetFeatureInfo request HTTP KVP encoding example....53
8.3.3 GetFeatureInfo HTTP KVP encoding response....53
8.3.4 GetFeatureInfo HTTP KVP encoding response example....53
8.4 Exceptions in HTTP KVP encoded operations....54
9 WMTS using SOAP encoding....54
9.1 GetCapabilities....54
9.1.1 GetCapabilities request SOAP encoding....54
9.1.2 GetCapabilities request SOAP encoding example....54
9.1.3 GetCapabilities SOAP encoding response....54
9.1.4 GetCapabilities SOAP encoding response example....55
9.2 GetTile....55
9.2.1 GetTile request SOAP encoding....55
9.2.2 GetTile request SOAP encoding example....55
9.2.3 GetTile SOAP encoding response....56
9.2.4 GetTile SOAP encoding response example....56
9.3 GetFeatureInfo....57

Copyright © 2010 Open Geospatial Consortium Inc.

9.3.1 GetFeatureInfo request SOAP encoding....57
9.3.2 GetFeatureInfo request SOAP encoding example....57
9.3.3 GetFeatureInfo SOAP encoding response....57
9.3.4 GetFeatureInfo SOAP encoding response example....58
9.4 Exceptions in SOAP encoding....59

10 WMTS using RESTful....60
10.1 ServiceMetadata resource (mandatory in resource oriented architectural style)....61
10.1.1 GetResourceRepresentation request....61
10.1.2 GetResourceRepresentation request example....61
10.1.3 ServiceMetadata representation....61
10.1.4 ServiceMetadata representation example....61
10.1.5 GetResourceRepresentation exception....61
10.2 Tile resource (mandatory in resource oriented architectural style)....62
10.2.1 GetResourceRepresentation request....62
10.2.2 GetResourceRepresentation request example....64
10.2.3 Tile representation....64
10.2.4 Tile representation example....64
10.2.5 GetResourceRepresentation exception....65
10.3 FeatureInfo resource (optional in resource oriented architectural style)....65
10.3.1 GetResourceRepresentation request....65
10.3.2 GetResourceRepresentation request example....68
10.3.3 FeatureInfo representation....68
10.3.4 FeatureInfo representation as an XML document example....68
10.3.5 GetResourceRepresentation exception....68

11 Recommendations to improve interoperability and performance....68
11.1 Server and Client support for KVP, SOAP and RESTful....68
11.2 A standard set of scales....69
11.3 A standard image format and FeatureInfo document response....69
11.4 Number of TileMatrixSets and TileMatrixSetLimits....69
11.5 Cacheble resources....69

Annex A (normative) Abstract test suite....71
A.1 Introduction....71
A.2 Client test module....72
A.3 Server test module....73

Annex B (normative) XML Schema Documents....86

Annex C (informative) UML model....88
C.1 Introduction....88
C.2 UML packages....89
C.3 WMTS Service package....90
C.4 WMTS Get Capabilities, Contents and Themes packages....91
C.5 WMTS Get Tile package....94
C.6 WMTS Get Feature Info package....96

 Copyright © 2010 Open Geospatial Consortium Inc.

Annex D (informative) Example XML documents 98
D.1 Introduction 98
D.2 ServiceMetadata response example....98

Annex E (informative) Well-known scale sets 102
E.1 GlobalCRS84Scale (urn:ogc:def:wkss:OGC:1.0:GlobalCRS84Scale)....102
E.2 GlobalCRS84Pixel (urn:ogc:def:wkss:OGC:1.0:GlobalCRS84Pixel)....103
E.3 GoogleCRS84Quad (urn:ogc:def:wkss:OGC:1.0:GoogleCRS84Quad) 104
E.4 GoogleMapsCompatible (urn:ogc:def:wkss:OGC:1.0:GoogleMapsCompatible)....105

Annex F (normative) WSDL description of the service 106
F.1 General 106
F.2 WSDL Publication 106
F.3 Abstract and concrete WSDL documents 107
F.4 Abstract WMTS WSDL document 107
F.5 Concrete WMTS WSDL document 109
F.6 Concrete WMTS WSDL document example 109

Annex H (informative) Pseudocode....112
H.1 From BBOX to tile indices....112
H.2 From tile indices to BBOX....112

Bibliography 114

Figures

Page

Figure 1 — WMTS interface UML diagram 7
Figure 2 — Tile Space 9
Figure 3 — Tile Matrix Set representation 10
Figure 4 — ServiceMetadata UML model 12
Figure 5 — Contents UML model 16
Figure 6 — Layer UML model 18
Figure 7 — Optional TileMatrix Limits 25
Figure 8 — TileMatrixSet UML model 25
Figure 9 — Themes UML model 29
Figure 10 — GetTile operation request UML class diagram 41
Figure 11 — GetFeatureInfo operation request UML class diagram 46
Figure 12 — GetTile response example 52
Figure 13 — URLTemplate for tile UML class diagram 62
Figure 14 — URLTemplate for the FeatureInfo UML class diagram 65

Copyright © 2010 Open Geospatial Consortium Inc.

Figure C.1 — WMTS interface UML diagram 88
Figure C.2 — WMTS interface package diagram 89
Figure C.3 — WMTS Service package class diagram 90
Figure C.4 — Get Capabilities package class diagram, part 1 91
Figure C.5 — Get Capabilities package class diagram, part 2 92
Figure C.6 — Get Capabilities package class diagram, part 3 93
Figure C.7 — Get Tile package class diagram 94
Figure C.8 — Get Feature Info package class diagram 96

Tables

Page

Table 1 — Contents of data dictionary tables 5
Table 2 —ServiceMetadata sections 13
Table 3 — Parts of the Capabilities data structure 13
Table 4 — Values of OperationsMetadata section parameters 14
Table 5 — Parts of the Contents section 17
Table 6 — Parts of Layer data structure 19
Table 7 — Parts of Style data structure 20
Table 8 — Parts of LegendURL data structure 21
Table 9 — Parts of Dimension data structure 22
Table 10 — Parts of TileMatrixSetLink data structure 23
Table 11 — Parts of TileMatrixSetLimits data structure 24
Table 12 — Parts of TileMatrixLimits data structure 24
Table 13 — Parts of TileMatrixSet data structure 26
Table 14 — Parts of TileMatrix data structure 27
Table 15 — Parts of the Themes section 29
Table 16 — Parts of Theme data structure 29
Table 17 — Parameters in GetCapabilities operation request 36
Table 18 — Meaning of section name values 37
Table 19 — Implementation of parameters in GetCapabilities operation request 37
Table 20 — Exception codes for GetCapabilities operation 38
Table 21 — HTTP exception codes and meanings on GetCapabilities operation 39
Table 22 — Parameters in GetTile operation request 42
Table 23 — Exception codes for GetTile operation 43
Table 24 — HTTP exception codes and meanings on GetTile operation 43

 Copyright © 2010 Open Geospatial Consortium Inc.

Table 25 — Parameters in GetFeatureInfo operation request 47
Table 26 — Exception codes for GetFeatureInfo operation..... 48
Table 27 — HTTP exception codes and meanings on GetFeatureInfo operation 48
Table 28 — Procedure oriented architectural style operation request encoding 49
Table 29 — GetTile operation request URL parameters..... 51
Table 30 — GetFeatureInfo operation request URL parameters..... 52
Table 31 — Parts of the URLTemplate data structure for tiles..... 62
Table 32 — URL template variables and possible values for tile 63
Table 33 — Parts of the URLTemplate data structure for FeatureInfo 65
Table 34 — URL template variables and possible values for FeatureInfo..... 66
Table C.1 — Mapping of UML TileRequest attributes and HTTP GetTile request parameters... 95
Table C.2 — Mapping of UML FeatureInfoRequest attributes and HTTP GetFeatureInfo request parameters 97
Table E.1 — Definition of Well-known scale set GlobalCRS84Scale 102
Table E.2 — Definition of Well-known scale set GlobalCRS84Pixel..... 103
Table E.3 — Definition of Well-known scale set GoogleCRS84Quad..... 104
Table E.4 — Definition of Well-known scale set GoogleMapsCompatible 105

Copyright © 2010 Open Geospatial Consortium Inc.

i. Preface

This document defines an OGC standard for a Web Map Tile Service (WMTS) interface standard. A WMTS enabled server application can serve map tiles of spatially referenced data using tile images with predefined content, extent, and resolution.

Suggested additions, changes and comments on this standard are welcome and encouraged. Such suggestions may be submitted using the online change request form on OGC web site: http://portal.opengeospatial.org/public_ogc/change_request.php

ii. Standard verbs and usages

This document uses the standard verbs defined in subclause 5.3 of [OGC 06-121r3], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards that are also included in subclause 6.1 of [OGC 06-135r7] Policy Directives for Writing and Publishing OGC Standards. In particular, the word "SHALL" (in capital letters and not "must") is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

iii. Submitting organizations

The following organizations submitted this document to the Open Geospatial Consortium Inc.

Autonomous University of Barcelona

CREAF

CubeWerx Inc.

iv. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization	E-mail Address
Joan Masó	UAB-CREAF	joan.maso@uab.cat
Keith Pomakis	CubeWerx Inc.	pomakis@cubewerx.com
Núria Julià	UAB-CREAF	n.julia@creaf.uab.cat

Copyright © 2010 Open Geospatial Consortium Inc.

v. Revision history

Date	Release	Editor	Primary clauses modified	Description
2007-05-08	07-057	Joan Masó and Núria Julià		First draft as a discussion paper in the WMS.RWG
2007-06-15	07-057r1	Joan Masó		Corrections proposed mainly by Dimitri Monie and Benjamin Chartier.
2007-08-14	07-057r2	Keith Pomakis		Merged OGC 07-057r1 and OGC 07-085r1 into a unified document.
2007-11-13	07-057r3	Keith Pomakis		Now proposes Web Map Tiling Service as a separate service rather than as a profile of WMS; introduced GetLegendGraphic and GetAlternateSources operations.
2008-03-25	07-057r4	Keith Pomakis		Several modifications proposed mainly by Chuck Morris and Joan Masó.
2008-06-15	07-057r5	Joan Masó		Reformated as a Standards document. GetLegendGraphic and GetAlternateSources operations were removed. RESTful was introduced. Major revisions by Benjamin Chartier and Keith Pomakis.
2009-01-25		Joan Masó		Many considerations in Keith Pomakis OGC 09-006 *OWS-6-DSS Engineering Report – SOAPXML and REST in WMTS* have been included.
2009-02-26	07-057r6	Joan Masó		Added WSDL annex F and general minor corrections.
2009-08-11	07-057r7	Joan Masó		CRs in RFC period have been incorporated. URL template has been introduced. Name changes from "Web Map Tiling Service" to "Web Map Tile Service". Major revision by Nuke Goldstein, Herve Caumont, Satish Sankaran, Lacey Sharpe and Xavier Pons.
2010-02-01		Adrian Custer and Joan Masó		Adrian Custer deeply detailed revision on grammar and general consistency.
2010-02-14		Carl Reed		Additional edits in preparation for publication as an OGC standard.
2010-03-09		Adrian Custer and Joan Masó		Grammar and general consistency last details. MIME type correction following 09-144r1.

Copyright © 2010 Open Geospatial Consortium Inc.

vi. Changes to the OGC Abstract Specification

The OpenGIS® Abstract Specification does not require changes to accommodate the technical contents of this standard.

vii. Changes to OpenGIS® Implementation Standards

This document defines an OGC implementation standard called "OpenGIS® Web Map Tile Service Implementation Standard". This standard references the OGC Web Services Common Specification, version 1.1.0 (with Corrigendum 1). No other implementation standard should be affected.

viii. Future work

Add support for OWS Common 1.2 (approved late 2009). Several additions planned in OWS Common 1.2 [06-121r8] could potentially impact this standard and need to be evaluated.

Follow future guidance for resource oriented architectural (ROA) style that is expected to be defined in future versions of OWS Common.

 Copyright © 2010 Open Geospatial Consortium Inc.

Foreword

The Web Map Tile Service (WMTS) described in this standard builds on earlier efforts to develop scalable, high performance services for web based distribution of cartographic maps. WMTS is inspired by the OSGeo Tile Map Service Specification (available at http://wiki.osgeo.org/index.php/Tile_Map_Service_Specification). The team that worked on this standard also considered similar initiatives, such as Google maps and NASA OnEarth. This OGC standard includes both resource (RESTful approach) and procedure oriented architectural styles (KVP and SOAP encoding) in an effort to harmonize this interface standard with the OSGeo specification.

WMTS complements earlier efforts to develop services for the web based distribution of cartographic maps. The OGC WMTS provides a complementary approach to the OGC Web Map Service (WMS) for tiling maps. **WMS** focuses on rendering **custom maps** and is an ideal solution for dynamic data or custom styled maps (combined with the OGC Style Layer Descriptor (SLD) standard). **WMTS** trades the flexibility of custom map rendering for the scalability possible by serving of **static** data (base maps) where the bounding box and scales have been constrained to discrete **tiles**. The fixed set of tiles allows for the implementation of a WMTS service using a web server that simply returns existing files. The fixed set of tiles also enables the use of standard network mechanisms for scalability such as distributed cache systems.

This standard has been structured as a stand alone standard (relying on OpenGIS Web Service Common Implementation Specification OGC 06-121r3 as a base document) but shares many concepts with the WMS 1.3.0.

This document replaces any previous versions of OGC 07-057 that were released as OGC Discussion Papers.

This document includes 7 annexes; Annexes A, B and F are normative, and Annexes C, D E and G are informative.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Copyright © 2010 Open Geospatial Consortium Inc.

Introduction

This Web Map Tile Service (WMTS) Implementation Standard provides a standard based solution to serve digital maps using predefined image tiles. The service advertises the tiles it has available through a standardized declaration in the ServiceMetadata document common to all OGC web services. This declaration defines the tiles available in each layer (*i.e.* each type of content), in each graphical representation style, in each format, in each coordinate reference system, at each scale, and over each geographic fragment of the total covered area. The ServiceMetadata document also declares the communication protocols and encodings through which clients can interact with the server. Clients can interpret the ServiceMetadata document to request specific tiles.

The WMTS standard complements the existing Web Map Service standard of the OGC. The WMS standard focuses on flexibility in the client request enabling clients to obtain exactly the final image they want. A WMS client can request that the server creates a map by overlaying an arbitrary number of the map layers offered by the server, over an arbitrary geographic bound, with an arbitrary background color at an arbitrary scale, in any supported coordinate reference system. The client may also request that the map layers be rendered using a specific server advertised style or even use a style provided by the client when the WMS server implements the OGC Styled Layers Descriptor (SLD) standard. However, all this flexibility comes at a price: server image processing must scale with the number of connected clients and there is only limited potential to cache images between the server and client since most images are different.

As web service clients have become more powerful, it has become possible to consider an alternative strategy which forces the clients to perform image overlays themselves and which limits the clients to requesting map images which are not at exactly the right position thereby forcing the clients to mosaic the tiles obtained from the server and clip the set of tiles into a final image. This restriction of image requests to a fixed, predefined set allows for servers to scale based on communication processing abilities rather than image processing abilities because servers can prerender some or all of their images and can use image caching strategies. The fixed set of images also enables network providers to cache images between the client and the server, reducing latency and bandwidth use. Popular, non standardized, commercial implementations of this approach, such as Google Maps, Microsoft Virtual Earth and Yahoo! Maps have already shown that there are clear performance benefits to adopting this methodology.

Some WMS servers have already embarked on this road, developing their own tiling structures built by constraining WMS GetMap requests to a fixed set and then advertising those constraints in their service metadata. Although this mechanism enables those servers to scale as just described, the tiling structure and the advertising and discovery mechanisms are not standardized. That unfortunately limits interoperability and forces developers to build, for each server, special clients that can understand the server advertised constraints and limit the WMS GetMap requests issued by the client to exactly

 Copyright © 2010 Open Geospatial Consortium Inc.

the requests understood by the particular server. This WMTS standard offers a standardized approach to declaring the images which a client can request from a server, enabling a single type of client to be developed for all servers. While developing a profile of WMS was initially considered, limiting a WMS in the ways important to allow efficient access to cacheable tiles proved awkward while forcing implementors to read both a standard and a profile seemed less efficient than developing this stand alone specification.

This standard specifies WMTS in two stages. First, an abstract specification describes the semantics of the resources offered by the servers and requested by the client. This abstract definition specifies the semantics of the **ServiceMetadata** document, of the **Tile** images or representations, and of the optional **FeatureInfo** documents providing descriptions of the maps at specific locations. Second, this standard specifies several different concrete exchange mechanisms between clients and servers in two different architectural styles. The standard defines the **GetCapabilities**, **GetTile** and optional **GetFeatureInfo** operations for procedure oriented architectural style based approaches using several different message encodings, including messages encoded using Key-Value Pairs (KVP), XML messages, or XML messages embedded in SOAP envelopes. The standard also defines the request mechanisms and endpoint publishing strategy to enable a resource oriented architectural style based on web based URL endpoints allowing clients to simply request the ServiceMetadata, Tile, and FeatureInfo resources as documents.

This resource oriented architecture style is new to the OGC but offers key advantages in ease of deployment, scalability and network effects of OGC web services. The RESTful pattern provides the ability to set up conformant WMTS servers simply. If all the images are prerendered, a WMTS server could even be created using no image processing logic at all but relying only on a normal web server to return the static ServiceMetadata XML document and provide the image tile files. This is important for deployment purposes as many Internet service providers (especially the free ones) allow web pages and static content hosting but do not allow using CGI, ASP, or more advanced applications for security reasons. The RESTful approach therefore enables small organizations to provide geographic data using readily available services or simple web server configurations. This approach also scales dramatically since the issues of serving fixed resources in high volumes have been continuously tackled over the past decades. Finally, this approach can benefit from network scaling effects since the images are considered by the HTTP protocol to be standard web resources and network providers can leverage their existing technologies to improve the flow of those resources to requesting clients.

Copyright © 2010 Open Geospatial Consortium Inc.

OpenGIS® Web Map Tile Service Implementation Standard

1 Scope

This OGC® document specifies an interface standard called "OpenGIS® Web Map Tile Service Implementation Standard" (WMTS).

This OGC® document is applicable to servers and clients that can serve and consume rendered tile maps. It can be combined with other OGC standards and also integrated with the emerging RESTful applications and "mash-ups".

2 Compliance

Compliance with this standard SHALL be checked using all the relevant tests specified in Annex A (normative).

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

ISO 19105:2000, *Geographic information — Conformance and Testing*

OGC 06-121r3, *OpenGIS® Web Services Common Specification, version 1.1.0 with Corrigendum 1, Arliss Whiteside, ed.,*

NOTE This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

W3C *SOAP Version 1.2 Part 1: Messaging Framework, W3C Recommendation* 24 June 2003,

W3C *SOAP 1.2 Attachment Feature*, W3C Working Group Note 8 June 2004,

W3C *Web Services Description Language (WSDL) 1.1*, W3C Note 15 March 2001,

In addition to this document, this standard includes, as normative, several XML Schema Document files as described in Annex B.

Copyright © 2010 Open Geospatial Consortium Inc.

4 Terms and definitions

For the purposes of this standard, the definitions specified in clause 4 of the OWS Common Implementation Specification [OGC 06-121r3] and SHALL apply. In addition, the following terms and definitions apply.

4.1
coordinate reference system
coordinate system that is related to the real world by a datum

4.2
coordinate system
set of mathematical rules for specifying how coordinates are to be assigned to points

4.3
feature
abstraction of a real world phenomenon

4.4
feature info
information related to a particular pixel of a map that refers to the geographic data portrayed on that area

4.5
layer
basic unit of **geographic information** that may be requested as a **map** from a **server**

4.6
map
portrayal of geographic information as a digital image file suitable for display on a computer screen

4.7
portrayal
graphical presentation of information to humans

4.8
procedure oriented architectural style
platform-independent design approach that is focused on operations, their parameters and their results, that can be defined in an abstract level specification. Concrete platform-dependent specifications can be derived from the abstract level, allowing, for example, KVP or SOAP messaging.

4.9
resource oriented architectural style
platform-independent design approach that is focused on resources, representations and actions, that can be defined in an abstract level specification. Concrete platform-dependent specifications can be derived form the abstract level, allowing, for example, a RESTful architecture.

 Copyright © 2010 Open Geospatial Consortium Inc.

4.10
theme
a group layers that can be nested hierarchically

4.11
tile
a rectangular pictorial representation of geographic data, often part of a set of such elements, covering a spatially contiguous extent and sharing similar information content and graphical styling, which can be uniquely defined by a pair of indices for the column and row along with an identifier for the tile matrix.

4.12
tile matrix
a collection of tiles for a fixed scale

4.13
tile matrix set
a collection of tile matrices defined at different scales

5 Conventions

5.1 Abbreviated terms

Most of the abbreviated terms listed in subclause 5.1 of the OWS Common Implementation Specification [OGC 06-121r3] apply to this document, plus the following abbreviated terms.

ASP	Active Server Pages
CGI	Common Gateway Interface
JPEG	Joint Photographic Experts Group (image format)
JPIP	JPEG 2000 Interactive Protocol
PNG	Portable Network Graphics (image format)
REST	Representational State Transfer
SLD	Styled Layer Descriptor
SOAP	Simple Object Access Protocol
WMTS	Web Map Tile Service
WSDL	Web Services Description Language

Copyright © 2010 Open Geospatial Consortium Inc.

5.2 UML notation

Most diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in subclause 5.2 of OWS Common [OGC 06-121r3].

5.3 Used parts of other documents

This document uses significant parts of document [OGC 06-121r3], herein referred to as "OWS Common". To reduce the need to refer to that document, this document copies some of those parts with small modifications. In tables and figures, to indicate those parts to readers of this document, the largely copied parts are shown with a light grey 15% background.

5.4 Platform-neutral and platform-specific standards

As specified in clause 10 of OGC Abstract Specification Topic 12 “OpenGIS Service Architecture” (which contains ISO 19119), this document includes both Distributed Computing Platform-neutral and platform-specific standards. This document first specifies the resources in each operation request and response in platform-neutral fashion. This is done using a table for each data structure, which lists and defines the parameters and other data structures contained. These tables serve as data dictionaries for the UML model in Annex C, and thus specify the UML model data type and multiplicity of each listed item.

NOTE 1 Platform-neutral standards are contained in clause 7.

The specified platform-neutral data could be encoded in many alternative ways, each appropriate to one or more specific Distributed Computing Platform. This document currently specifies encodings appropriate for HTTP GET transfer of operations requests (using KVP or RESTful encodings), and for HTTP POST transfer of operations requests (using XML or SOAP encodings). However, the same operation requests and responses (and other data) could be encoded for other specific computing platforms, including HTTP POST transfer of raw XML requests.

NOTE 2 Platform-specific standards for KVP, SOAP and RESTful are contained in clause 8, 9 and 10 respectively.

5.5 UML graphical and table representations

The UML model data is specified herein in a series of tables, called data dictionary tables. The contents of the columns in these tables are described in Table 1.

 Copyright © 2010 Open Geospatial Consortium Inc.

Table 1 — Contents of data dictionary tables

Column title	Column contents
Names (left column)	Two names for each included parameter or association (or data structure). The first name is the UML model attribute or association role name. The second name is the XML encoding of the parameter name. It is shown in monospaced font. Some names in the tables may appear to contain spaces, but no names contain spaces.
Definition (second column)	Specifies the definition of this parameter (omitting un-necessary words such as "a", "the", and "is"). If the parameter value is the identifier of something, not a description or definition, the definition of this parameter reads something like "Identifier of …".
Data type and values (third column)	The first item is the data type used for this parameter, using data types or a data structure appropriate in a UML model, in which this parameter is a named attribute of a UML class. Alternately, the first item can identify the data structure (or class) referenced by this association, or references a separate table used to specify the contents of that class (or data structure). The second item indicates the source of values for this parameter, the alternative values, or other value information, unless the values are quite clear from other listed information.
Multiplicity and use (right or fourth column)	The first item specifies the multiplicity and optionally of this parameter in this data structure, either "One (mandatory)", "One or more (mandatory)", "Zero or one (optional)", or "Zero or more (optional)". The second item specifies how any multiplicity other than "One (mandatory)" will be used.

When the data type used for this parameter, specified in the third column of such a table is an enumeration or code list, all the values specified by this document are listed, together with the meaning of each value. When this information is extensive, these values and meanings are specified in a separate table that is referenced in the third column of the table row where the parameter is defined.

NOTE Several parameters have their data type specified in the third table column as "Character String type, not empty". In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, an XML type which does not require that these strings not be empty. Nonetheless, the injunction of the table SHALL prevail and the element SHALL not be empty

The contents of these data dictionary tables are normative, including any table footnotes. Particularly, the "Multiplicity and use" columns in Table 6 through Table 16 in OWS Common [OGC 06-121r3], and in Table 2, 3 and Table 5 through Table 16 of this document, specify the optionality of each listed parameter and data structure in the ServiceMetadata document. Also, "Multiplicity and use" columns of this document in Table 22 specify the optionality of each listed parameter and data structure in the GetTile operation request and in Table 25 specify the optionality of each listed parameter and data structure in the GetFeatureInfo operation request. All the "mandatory" parameters and data structures SHALL be implemented by all WMTS clients, using a specified value(s). Similarly, all the "mandatory" parameters and data structures SHALL be implemented by all WMTS servers, checking that each request parameter or data structure is received with the specified value(s). All the "optional" parameters and data structures in the operation requests SHOULD be implemented by all WMTS clients using

Copyright © 2010 Open Geospatial Consortium Inc.

specified values, for each implemented layer to which that parameter or data structure applies. Similarly, all the "optional" parameters and data structures SHALL be implemented by all WMTS servers, for each implemented layer to which that parameter or data structure is declared to apply by the server in the ServiceMetadata document.

6 WMTS overview

The goal of providing a WMTS enabled service is to be performance oriented and scalable. Therefore, servers must be able to return tiles quickly. A good way to achieve that is to use locally stored pre-rendered tiles that will not require any image manipulation or geo-processing. Server developers will decide if pre-rendered tiles will be generated in a previous tile-preparation process or generated on the fly utilizing a caching mechanism. With tile-based mapping it is important that the server will be able to handle asynchronous access to tiles as most clients will simultaneously query for multiple tiles to fill a single view.

The purpose of a WMTS service is to serve maps divided in individual tiles.

The WMTS interface allows a client to receive three types of resources either in response to a resource request in the resource oriented architectural style or in response to an operation in the procedure oriented architectural style. Those resources and operations are:

a) A **ServiceMetadata** resource (in response to a GetCapabilities operation for the procedure oriented architectural style) (required implementation by servers) – It describes the abilities and information holdings of the specific server implementation. In procedure oriented architectural style this operation also supports negotiation of the standard version being used for client-server interactions.

b) A **tile** resource (in response to a GetTile operation for the procedure oriented architectural style) (required implementation by servers) – It shows a fragment of a map representation of a layer.

c) A **FeatureInfo** resource (in response to a GetFeatureInfo operation for the procedure oriented architectural style) (optional implementation by servers) – It provides information about the features located at a particular pixel of a tile map, in a similar way to the WMS GetFeatureInfo operation, by providing, for example, the thematic attribute name and value pairs in textual form.

These operations have many similarities to other OGC Web Services (OWS), including the Web Map Service (WMS), Web Feature Service (WFS), and Web Coverage Service (WCS). Many of the aspects of this WMTS interface that are shared in common with other OWSs are specified in the OpenGIS® Web Services Common Implementation Specification [OGC 06-121r3]. Many of these common aspects are included normatively by reference to that document, instead of being repeated in this standard.

 Copyright © 2010 Open Geospatial Consortium Inc.

Figure 1 is a simple UML diagram summarizing the WMTS interface for the procedure oriented architectural style. This class diagram shows that the WMTS interface class inherits the getCapabilities operation from the OGCWebService interface class, and adds the getTile and getFeatureInfo operations. (This capitalization of names uses the OGC/ISO profile of UML.) A more complete UML model of the WMTS interface is provided in Annex C (informative).

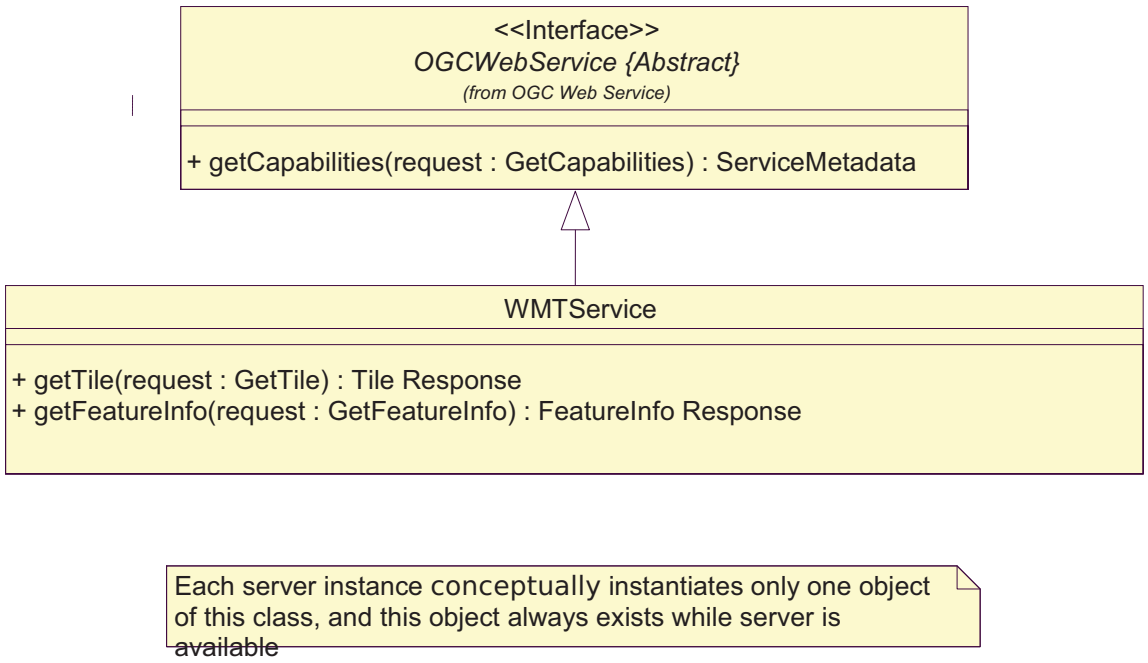


Figure 1 — WMTS interface UML diagram

NOTE In this UML diagram, the request and response for each operation is shown as a single parameter that is a data structure containing multiple lower-level parameters, which are discussed in subsequent clauses. The UML classes modeling these data structures are included in the complete UML model in Annex C.

The WMTS serves a single tile of a single layer of a map. Unlike WMS, there is no specified way to request a server to combine and return a map tile with information coming from more than one layer in a single fetching process. WMTS clients that want to show a combination of layers must make independent requests for the layer tiles and then combine or overlay the responses. Also bounding boxes and scales of these WMTS tiles are constrained to a discrete set of values.

6.1 Tile matrix set – the geometry of the tiled space

In a tiled map layer, the representation of the space is constrained in a discrete set of parameters. A tile matrix set defines these parameters. Each tile matrix set contains one or more "tile matrices" defining the tiles that are available for that coordinate reference system. Each tile matrix specifies:

Copyright © 2010 Open Geospatial Consortium Inc.

a) The scale of the tiles as a scale denominator.

The scale denominator is defined with respect to a "standardized rendering pixel size" of 0.28 mm × 0.28 mm (millimeters). The definition is the same used in WMS 1.3.0 [OGC 06-042] and in Symbology Encoding Implementation Specification 1.1.0 [05-077r4]. Frequently, the true pixel size is unknown and 0.28 mm is a common actual size for current displays.

b) The width and height of each tile in pixels.

c) The top left (minimum x, maximum y) corner of the bounding box of the tile matrix (*i.e.*, the CRS coordinates of the top left corner of the top left pixel of the top left tile).

d) The width and height of the tile matrix in tile units (*i.e.*, number of tiles).

The number of tile matrix sets that a WMTS server serves for a particular layer is:

$$nTileMatrices \times nTiledStyles \times nTiledFormats$$

if no dimensions are defined or:

$$nTileMatrices \times nTiledStyles \times nTiledFormats \times nTiledDimensions$$

if dimensions are defined. The number of distinct tiles within each tile matrix of a tile matrix set (*i.e.*, for a particular scale within a tile-matrix set) is a product of:

$$matrixWidth \times matrixHeight$$

Each tile matrix set defines its own set of scale levels corresponding with the contained tile matrices. Each layer references one or more tile matrix sets. Although each layer could reference a different tile matrix set, it is likely that a server will offer many layers with the same tile matrix set reference.

A tile matrix set is composed of a collection of tile matrices, each one with a resolution optimized for a particular scale and identified by a tile matrix identifier (see figure 3). Each tile matrix set has an optional approximated bounding box but each tile matrix has an exact bounding box that is deduced indirectly from other parameters. Tile matrix bounding boxes at each scale will usually vary slightly due to pixel alignment, and it is important for the client and server to take this variation into account. Given the top left point of the tile matrix in CRS coordinates (tileMatrixMinX, tileMatrixMaxY), the width and height of the tile matrix in tile units (matrixWidth, matrixHeight), the width and height of a tile in pixels (tileWidth, tileHeight), the coefficient to convert the coordinate reference system (CRS) units into meters (metersPerUnit) and the scale (1:scaleDenominator), the bottom right corner of the bounding box of a tile matrix (tileMatrixMaxX, tileMatrixMinY) can be calculated as follows:

$$pixelSpan = scaleDenominator \times 0.28\ 10^{-3} / metersPerUnit(crs);$$

$$tileSpanX = tileWidth \times pixelSpan;$$

 Copyright © 2010 Open Geospatial Consortium Inc.

tileSpanY = tileHeight × *pixelSpan;*

tileMatrixMaxX = tileMatrixMinX + tileSpanX × *matrixWidth;*

tileMatrixMinY = tileMatrixMaxY - tileSpanY × *matrixHeight;*

The tile space therefore looks like this:

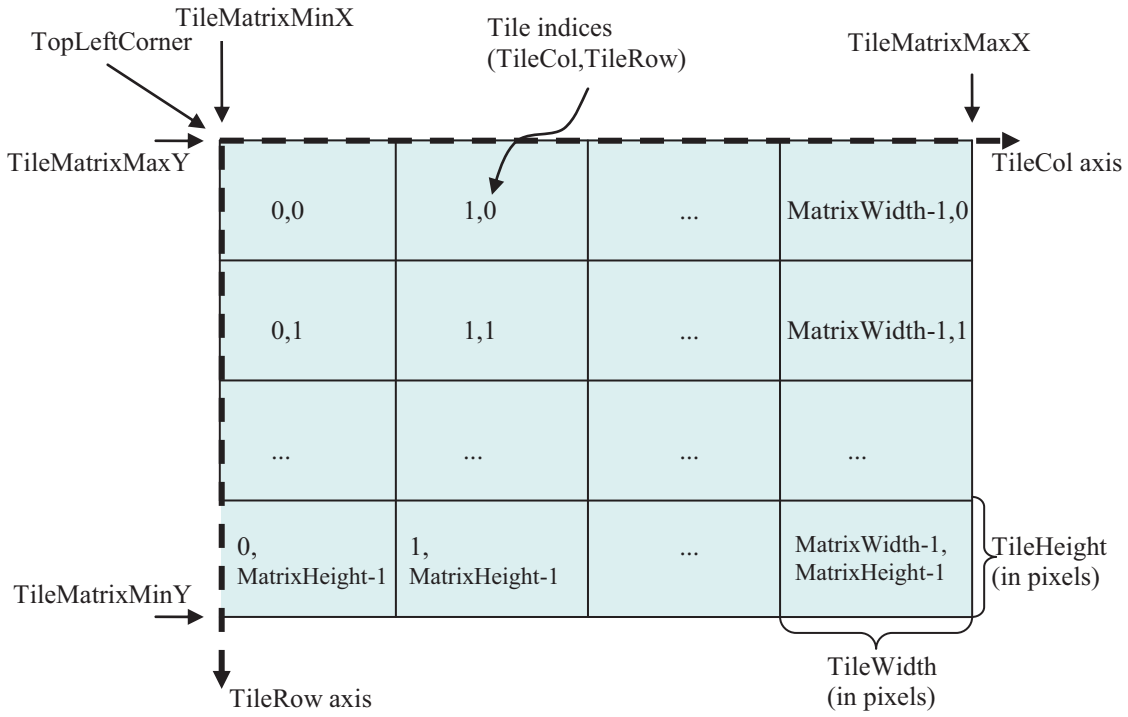


Figure 2 — Tile Space

Each tile in a tile matrix is identified by its TileCol and TileRow indices that have their 0,0 origin in the tile next to the top left corner of the tile matrix and that increases towards the right and towards the bottom respectively, as shown in figure 2. Annex H in this document includes pseudo code that illustrates the process for obtaining the tile indices that cover a bounding box rectangle and also the computation to get the CRS coordinates that bound a tile.

NOTE 1 Non-square pixels are not supported. This is different from WMS, which does allow non-square pixels (although many implementations fail to support this properly).

A tiled layer links to its tile matrix set through a tileMatrixSet URI that points to a TileMatrixSet section that completely defines it as previously explained. A layer can use a specific TileMatrixSet that describes a region adjusted to the actual content of this layer. In this case, the optional tileMatrixSetLimits section will not be used and changes in spatial extension of the layer can affect the minimum bounding box of the layer forcing to redefine the TopLeftCorner of each TileMatrix and that will end up changing the TileCol,TileRow indices thereby invalidating any previously cached tile. To

Copyright © 2010 Open Geospatial Consortium Inc.

overcome this problem a layer can optionally use a more generic TileMatrixSet that covers a bigger (or even global) area. In fact, that TileMatrixSet will define an area that could be covered by the layer in a future and could easily be shared for many layers in this server. To inform the client about the valid range of the TileCol and Tile Row indices a layer definition can optionally use the tileMatrixSetLimits section that specifies a minimum and a maximum that are limits of these indices for each TileMatrix. Any request outside these limits will result in a server exception (see Figure 6).

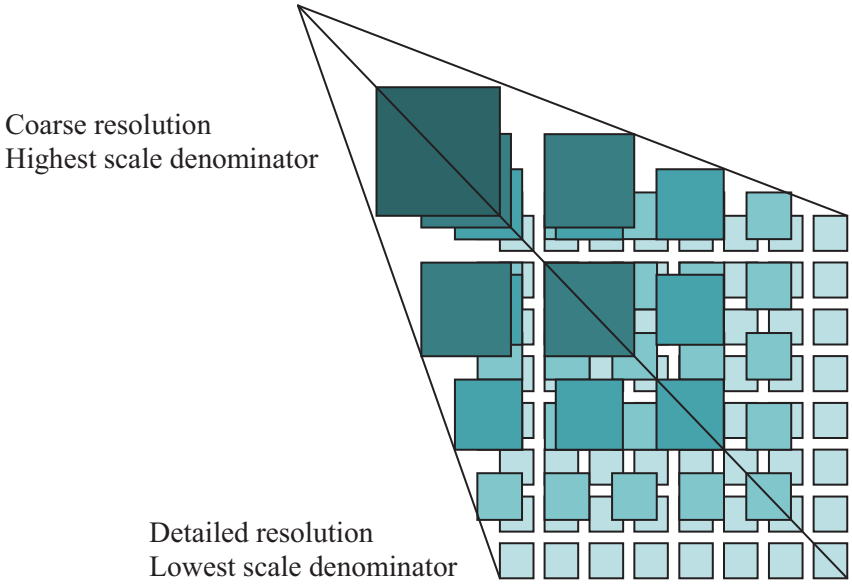


Figure 3 — Tile Matrix Set representation

In some other standards, this way of dividing the space is called *image pyramid* like in clause 11.6 of the KML 2.2 [OGC 07-147r2]. JPEG2000 (ISO/IEC 15444-1) and JPIP (ISO/IEC 15444-9) also use a similar division of the space called *resolution levels*. Nevertheless, in those cases the pyramid is self defined starting from the more detailed tile matrix that uses square tiles, and constructing tiles of the next scales by successively aggregating 4 tiles of the previous scale and so on. That approach involves a more rigid structure which has scales related by powers of two and tiles that perfectly overlap tiles on the inferior scale denominators. Since WMTS is more flexible, KML *superoverlays* or JPEG2000 based implementations can still use WMTS to describe their tile matrix sets and to serve tiles. Annex E.3 and E.4 describe scale sets related by powers of two.

Each of the WMTS procedure oriented architectural style operations and resource oriented architectural style resources are described in more detail in subsequent clauses.

NOTE 2 Clients and servers have to be careful when comparing floating numbers with tolerance (double precision, 16 digit numbers, has to be used).

6.2 Well-known scale sets

Since a WMTS server will serve its data in a limited number of coordinate systems and scales (because, unlike a WMS, it serves only pre-defined tiles), and since some simple

 Copyright © 2010 Open Geospatial Consortium Inc.

WMTS client will be unable to perform coordinate-system transformations or rescaling of tiles, the ability for a WMTS client to overlay tiles from one server on top of tiles from other servers will be limited unless there are some general agreements among WMTS servers as to what constitutes a common coordinate reference system and a common set of scales. Thus, this standard defines the concept of well-known scale sets. In order to increase interoperability between clients and servers it is recommended that many layers use a common set of scales in the same CRS that the target community agree to use.

A well-known scale set is a well-known combination of a coordinate reference system and a set of scales that a tile matrix set declares support for. Each tile matrix set references one well-known scale set. A client application can confirm that tiles from one WMTS server are compatible with tiles from another WMTS server merely by verifying that they declare a common well-known scale set. It may also be the case that a client application is limited to supporting a particular coordinate system and set of scales (*e.g.*, an application that overlays WMTS tiles on top of Google Maps tiles). In this situation, a client application can accept or reject a WMTS as being compatible merely by verifying the declared well-known scale set. Furthermore, the existence of well-known scale sets provides incentive for WMTS servers to support a well-known scale set, increasing the odds of compatibility with other WMTS sources. The informative Annex E provides several well-known scale sets and others could be incorporated in the future.

A tile matrix set conforms to a particular well-known scale set when it uses the same CRS and defines all scale denominators ranging from the largest scale denominator in the well-known scale set to some low scale denominator (in other words, it is not necessary to define all the lower scale denominators to conform to a well-known scale set).

NOTE 1 Well-known scale sets are technically not necessary for interoperability, since a client application can always examine the actual list of coordinate systems and scales available for each layer of a WMTS server in order to determine its level of compatibility with other WMTS servers. Well-known scale sets are merely a convenience mechanism.

7 WMTS Implementation model

This clause describes the WMTS resources that can be requested by a client from a server in either the procedure oriented architectural style or in the resource oriented architectural style. It also describes the procedure oriented architectural style operations in an abstract way; for KVP encoding, see clause 8 and for SOAP encoding, see clause 9. Resource oriented architectural style description and a RESTful implementation can be found in clause 10.

7.1 Service metadata

This subclause describes the ServiceMetadata document and the way in which it may be obtained using either a procedure oriented architectural style or a resource oriented architectural style.

Copyright © 2010 Open Geospatial Consortium Inc.

7.1.1 ServiceMetadata document

The ServiceMetadata document is the response document of a GetCapabilities request in procedure oriented architectural style or of a standard request to the right endpoint in a resource oriented architectural style. It is the entry point resource that represents the resources available on the service and communication requirements for the service.

7.1.1.1 ServiceMetadata document description

The ServiceMetadata document contains all the sections specified in Table 2, but partial documents can be requested containing only a subset of these sections. Depending on the values in the Sections parameter of the GetCapabilities operation request in the procedure oriented architectural style (see subclause 7.1.2.1), any combination of these sections can be requested and SHALL be returned when requested except if the service does not support requests for sub-sections of the ServiceMetadata document.

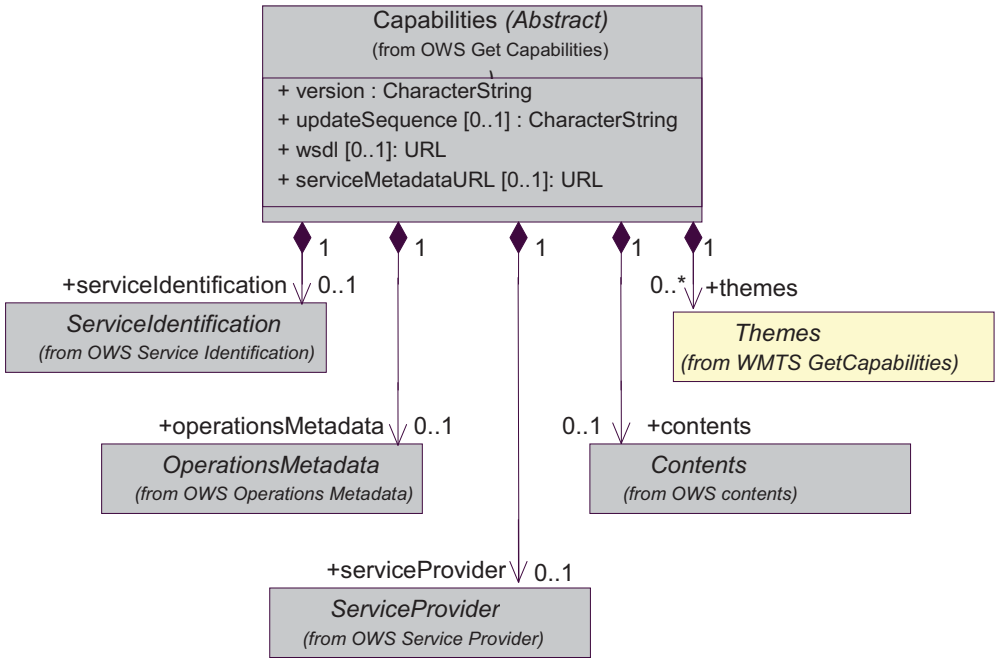


Figure 4 — ServiceMetadata UML model

Copyright © 2010 Open Geospatial Consortium Inc.

Table 2 —ServiceMetadata sections

Section name	Contents
ServiceIdentification	Metadata about this specific server. The schema of this section SHALL be the same as for all OWSs, as specified in subclause 7.4.3 and owsServiceIdentification.xsd of OWS Common [OGC 06-121r3].
ServiceProvider	Metadata about the organization operating this server. The schema of this section SHALL be the same for all OWSs, as specified in subclause 7.4.4 and owsServiceProvider.xsd of OWS Common [OGC 06-121r3].
OperationsMetadata	Metadata about the operations specified by this service in procedure oriented architectural style and implemented by this server, including the URLs for operation requests. The basic contents and organization of this section is the same as for all OWSs, as specified in subclause 7.4.5 and owsOperationsMetadata.xsd of OWS Common [OGC 06-121r3].
Contents	Metadata about the data served by this server. For the WMTS, this section SHALL contain data about layers and TileMatrixSets, as specified in Tables 5 through 14 below.
Themes	Metadata describing a theme hierarchy for the layers, as specified in Tables 15 and 16 below.

The ServiceIdentification and ServiceProvider sections are described on subclause 7.4.4 and 7.4.5 of OWS Common [OGC 06-121r3]. The OperationsMetadata, Contents and Themes sections are described in subclauses 7.1.2.1, 7.1.2.2 and 7.1.2.3 of this document.

In addition to these sections, each service metadata document SHALL include the mandatory "version" parameter and can optionally include "updateSequence" parameter specified in Table 9 in subclause 7.4.2 of OWS Common [OGC 06-121r3] and copied below. Finally, "WSDL" and "serviceMetadataURL" parameters are only needed for servers using specific encodings.

Table 3 — Parts of the Capabilities data structure

Names	Definition	Data type and values	Multiplicity and use
version `version`	Standard version for operation, in this case for GetCapabilities operation response	Character String type, not empty. Value is list of x.y.z "version" values. SHALL be "1.0.0"	One (mandatory)
updateSequence `updateSequence`	Service metadata document version, value is "increased" whenever any change is made in complete service metadata document	Character String type, not empty Values are selected by each server, and are always opaque to clients	Zero or one (optional) Omitted when parameter not supported by server
WSDL `WSDL`	Reference to a WSDL resource	URL type	Zero or more (optional) Only for SOAP encoding

Copyright © 2010 Open Geospatial Consortium Inc.

Names	Definition	Data type and values	Multiplicity and use
service MetadataURL `Service MetadataURL`	Reference to a ServiceMetadata resource	URL type	Zero or more (optional) Mandatory in resource oriented architectural style
serviceIdentification `Service Identification`	Metadata about this specific server	ServiceIdentification section, see Table 11 of OWS Common [OGC 06-121r3]	Zero or one (optional)
serviceProvider `ServiceProvider`	Metadata about the organization operating this server.	ServiceProvider section, see Table 12 of OWS Common [OGC 06-121r3]	Zero or one (optional)
operationsMetadata `Operations Metadata`	Metadata about the operations specified by this service	OperationsMetadata section, see Table 13 of OWS Common [OGC 06-121r3]	Zero or one (optional)
contents `Contents`	Metadata about the data served by this server.	Contents section, see Table 5	Zero or one (optional)
themes `Themes`	Metadata describing a theme hierarchy for the layers	Themes section, see Table 15	Zero or more (optional)

Parameters "version", and "updateSequence" are described in subclause 7.4.2 of OWS Common [OGC 06-121r3]. Parameters "WSDL" and "serviceMetadataURL" are described in Annex F2 and subclause 010.1.1 of this document.

7.1.1.1.1 *OperationsMetadata* section contents

The OperationsMetadata section is the same as for all OGC Web Services, as specified in subclause 7.4.6 and owsOperationsMetadata.xsd of OWS Common [OGC 06-121r3]. It is only relevant in the procedure oriented architectural style. The parameters are specified in Table 4 bellow. In Table 4, the “Name” column uses dot-separator notation to identify parts of a parent item. The “Value” column references an operation parameter, in this case an operation name, and the meaning of including that value is listed in the right column.

Table 4 — Values of OperationsMetadata section parameters

Name	Value	Meaning of parameter value
Operation.name	GetCapabilities	The GetCapabilities operation is implemented by this server.
	GetTile	The GetTile operation is implemented by this server.
	GetFeatureInfo	The GetFeatureInfo operation is implemented by this server.

In addition to the values listed in Table 4, there are many optional values of the “Name” attributes and “Value” parameters in the OperationsMetadata section, which MAY be

 Copyright © 2010 Open Geospatial Consortium Inc.

included when considered useful. Most of these attributes and parameters are for recording the domains of various operation parameters and quantities.

The Operation data type allows specifying distributed computing platform (DCP) parameters and the encoding of this DCP as a Constraint within the DCP parameter.

All WMTS servers operating in a procedure oriented architecture style and using HTTP SHALL specify with an ows:Constraint parameter the encodings that MAY be sent using HTTP GET or HTTP POST. Each operation can support more than one encoding and the set of supported encodings CAN be different for each operation (but this is discouraged since it is not usually expected).

All WMTS servers operating in a procedure oriented architecture style and using HTTP SHALL specify the message encodings that MAY be sent using HTTP GET transfer of operation requests. Specifically, an ows:Constraint parameter SHOULD be included, with “GetEncoding” as the value of the “name” attribute and specifying the values allowed:

a) The value “KVP” indicates that KVP encoding is allowed, when using HTTP GET transfer as specified in clause 8.

Also, all WMTS servers operating in a procedure oriented architecture style and using HTTP SHALL specify the message encodings that MAY be sent using HTTP POST transfer of operation requests. Specifically, an ows:Constraint parameter SHALL be included, with “PostEncoding” as the value of the “name” attribute and specifying the values allowed:

a) The value “SOAP” shall indicate that SOAP encoding is allowed, as specified in clause 9.

b) The value “XML” shall indicate that XML encoding is allowed (without SOAP message encapsulation).

c) The value “KVP” shall indicate that KVP encoding is allowed, when using HTTP POST transfer.

If the HTTP connection point URL is different for different encodings of the operation requests, the URL SHALL be specified in an ows:Constraint parameter in each Get or Post section. If the connection point URL is the same for all encodings of all operation requests, this ows:Constraint parameter SHALL be included in the OperationsMetadata section. The constraint names and values presented in this subclause are the actual exact names and values that SHALL be used for each encoding explained and are not just examples.

Resource oriented architecture style HTTP encodings SHALL not be described in the OperationsMetadata section. Instead, the service metadata document provided by servers operating in a resource oriented architectural style SHALL use ResourceURL and ServiceMetadataURL to indicate support for that architectural style, as is explained in clause 10.

Copyright © 2010 Open Geospatial Consortium Inc.

7.1.1.1.2 *Contents* section contents

The Contents section of a ServiceMetadata document contains metadata about the data served by this server. For the WMTS, this Contents section contains a general description of the layers available and descriptions of the extra dimensions, styles, image formats and tile matrix sets that apply to each layer. The Contents section SHALL include parameters as specified in Table 5 through Table 14.

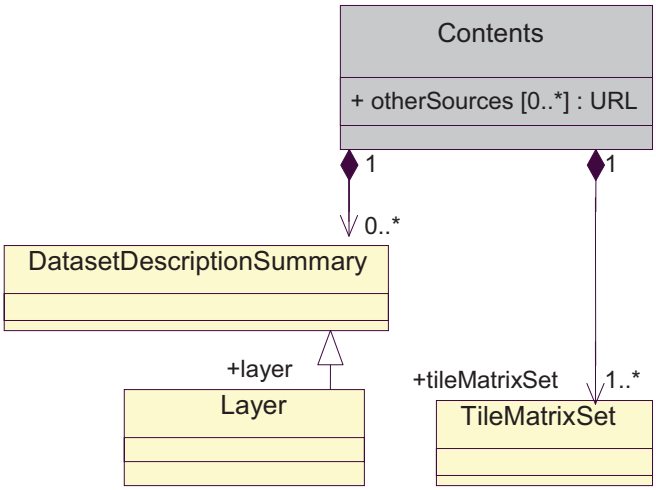


Figure 5 — Contents UML model

Table 5 through Table 12 define the components of the layer section and Table 13 through Table 14 define the components of the tile matrix set section of the ServiceMetadata document. The UML class diagram in Figure 5 provides a useful graphical view of the composition of the Contents section.

The Contents section, described in Table 5, contains a list of layers available on the server and a list of tileMatrixSets. Each layer links to a particular tileMatrixSet using a reference to a tileMatrixSet identifier. Layers are described in Table 6 with a name, a title, an abstract description, keywords, a WGS84BoundingBox, a tileMatrixSet reference, a supported image format list, an infoFormat list, a metadata URL document link, and an optional dimensions list. In addition, a layer has one or more map portrayal representations that are called styles. Each style is described with a Style section as detailed in Table 7 with a name, a title, an abstract and a list of legendURLs described in Table 8. Each legendURL provides an iconic representation of the layer in its style, suitable for display in a legend; it specifies the URL of the icon image, and optionally the width and height of the icon image and the range of scales for which the icon is appropriate, as described in Table 8. Optional dimensions of the layer are described in Table 9. Dimensions are described by an identifier, a title, an abstract, units and unit symbols, a list of possible values and a default value. Typical examples of dimension identifiers are "time", "elevation" and "band", but the service can define any other dimension property that exists in the multidimensional layer collection being served.

 Copyright © 2010 Open Geospatial Consortium Inc.

Table 5 — Parts of the Contents section

Names	Definition	Data type and values	Multiplicity and use
layer `Layer`	Metadata describing one top-level dataset available from this server	Layer data structure, see Table 6	Zero or more (optional) [a] One for each dataset available
otherSource `OtherSource`	Reference to another source of contents metadata	See CI_OnlineResource class in ISO 19115	Zero or more (optional)
tileMatrixSet `TileMatrixSet`	A description of the geometry of a tile cut	TileMatrixSet data structure, see Table 13	Zero or more (optional)

a SHALL be included unless Other Source parameter(s) are included and all this metadata is available from those sources.

The OtherSource parameter may reference one or more catalogue servers from which dataset metadata is available. This ability is expected to be used by servers with a very large number of datasets, for which searching a catalogue is more feasible than retrieving and then searching a very large ServiceMetadata XML document. When there is no Layer section in the Contents section of the ServiceMetadata document, the otherSource parameter SHALL reference one or more catalogue servers that contain current metadata summaries for all the datasets currently available from this WMTS server, with the metadata for each dataset referencing this WMTS server.

The UML class diagram in Figure 6 provides a useful graphical view of the Layer section with its properties, complex data types and dependencies.

Copyright © 2010 Open Geospatial Consortium Inc.

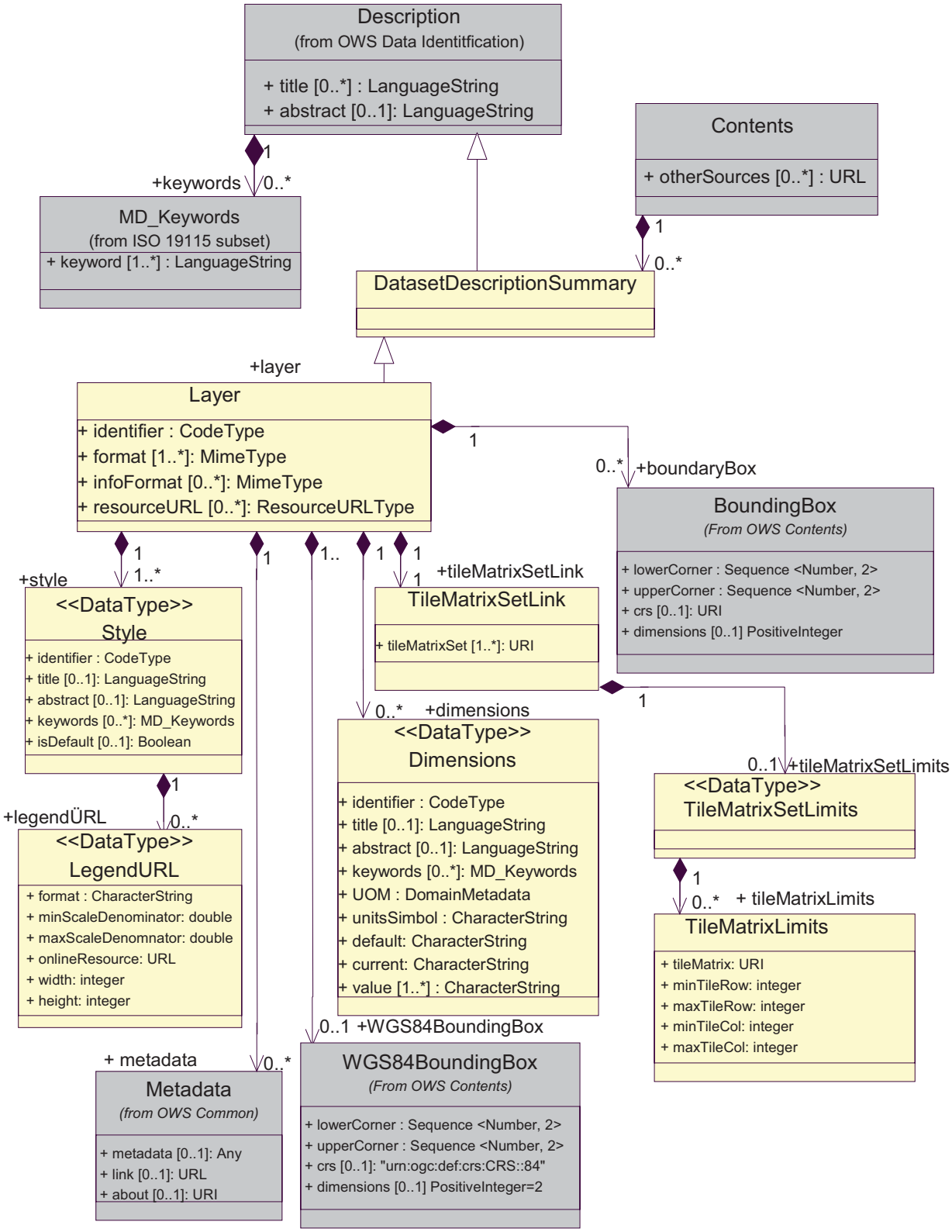


Figure 6 — Layer UML model

Copyright © 2010 Open Geospatial Consortium Inc.

Table 6 — Parts of Layer data structure

Names	Definition	Data type and values	Multiplicity and use
identifier [a] `Identifier`	An unambiguous reference to this layer, normally used by software [f]	ows:CodeType, as adaptation of MD_Identifier class ISO 19115	One (mandatory)
title [c] `Title`	Title of this layer, normally used for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented [e]
abstract [c] `Abstract`	Brief narrative description of this layer, normally available for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented
keywords [c] `Keywords`	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this dataset	MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
wgs84 BoundingBox `WGS84 BoundingBox`	Minimum bounding rectangle surrounding dataset, using WGS 84 CRS with decimal degrees and longitude before latitude [b]	WGS84Bounding Box data structure see subclause 10.2 of OWS Common [OGC 06-121r3]	Zero or one (optional)
boundingBox `BoundingBox`	Minimum bounding rectangle surrounding the layer, in the supported CRS [g]	BoundingBox data structure, see subclause 10.2 of OWS Common [OGC 06-121r3]	Zero or more (optional)
style `Style`	Description of the style that has been applied to this layer	Style data structure, see Table 7	One or more (mandatory)
format `Format`	Supported valid output formats for a tile request	ows:MimeType	One or more (mandatory)
infoFormat `InfoFormat`	Supported valid output formats for a FeatureInfo document request	ows:MimeType	Zero or more (optional) [d]
dimension `Dimension`	Extra dimensions for a tile and FeatureInfo resource requests	Dimension data structure, see Table 9	Zero or more (optional) One for each extra dimension available.

Copyright © 2010 Open Geospatial Consortium Inc.

Names	Definition	Data type and values	Multiplicity and use
metadata Metadata	Additional metadata about this dataset	ows:Metadata	Zero or more (optional) One for each useful metadata object
tileMatrixSet Link TileMatrix SetLink	Reference to a tileMatrixSet and limits	TileMatrixSetLink data structure, see Table 10	One or more (mandatory)
resourceURL ResourceURL	URL template to a tile or a FeatureInfo resource	URLTemplate data structure, see Table 31	Zero or more (optional) Include one or more in resource oriented architectural style

a This has the same meaning as "name" in WMS but has been replaced by "identifier" to harmonize with OWS Common

b This WGS84BoundingBox can be approximate, but SHOULD be as precise as practical.

c The multilingual scoping rules in subclause 10.7.3 of OWS Common [OGC 06-121r3] SHALL apply.

d If no infoFormats are specified, then the layer is not queryable (*i.e.*, a request for a FeatureInfo is not permitted for this layer)

e If no Title is specified, client may display the Identifier value instead.

f Layer identifies SHALL be unique (different) for each layer of this server

g It represents the area where this layer is represented. It could seem redundant with the bounding box of the tile matrix set but in complex cases that limits the area with data using tileMatrixLimits it is not so easy to calculate form the tile matrix set parameters.

The list of output formats SHOULD be chosen carefully. A long list of formats will improve interoperability with clients but will reduce the effectiveness of caching mechanisms. As a general rule, servers should use a short list, should avoid including redundant formats in the list and should use the formats recommended in subclause 11.3.

NOTE 1 In WMTS the list of supported output formats can be different for each layer, in contrast with WMS which specifies a shared single list of supported formats for all layers. WMTS layers have been given this ability because different layers may have different optimal formats. The use of a shared single list would force layers to be offered in all declared formats, reducing scalability and performance.

NOTE 2 The UML class diagram contained in the Annex C4 provides a useful graphical view of the contents of the Contents section listed in Tables 6 - 16.

Table 7 — Parts of Style data structure

Names	Definition	Data type and values	Multiplicity and use
identifier[a] Identifier	An unambiguous reference to this style, identifying a specific version when needed, normally used by software [d]	ows:CodeType, as adaptation of MD_Identifier class ISO 19115	One (mandatory)
title [b] Title	Title of this style, normally used for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented [c]

 Copyright © 2010 Open Geospatial Consortium Inc.

Names	Definition	Data type and values	Multiplicity and use
abstract [b] `Abstract`	Brief narrative description of this style, normally available for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented
keywords [b] `Keywords`	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this dataset	MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
legendURL `LegendURL`	Description of an image that represents the legend of the map	LegendURL data structure, see Table 8	Zero or more (optional) Include when available and useful
isDefault `isDefault`	The style that a client SHOULD select as the first choice (default style)	Boolean	Zero or one (optional) Default is "false" [e]

a This has the same meaning as "name" in WMS but has been replaced by "identifier" to harmonize with OWS Common.

b The multilingual scoping rules in subclause 10.7.3 of OWS Common [OGC 06-121r3] SHALL apply.

c If no Title is specified, client may display the Identifier value instead.

d Style identifies SHALL be unique (different) for each style of a particular layer

e Only one style per layer can have a "true" value

A WMTS ServiceMetadata document may include zero or more LegendURL elements to provide an image(s) of a legend relevant to each style of a layer. Clients can show this image to the user as a visual summary of the information rendered in the tiles. The legend image should clearly represent the symbols, lines and colors used in the portrayal of the tiles and their meanings. The legend image should not contain text that duplicates the title of the layer, because that information is known to the client and may be shown to the user by other means.

Table 8 — Parts of LegendURL data structure

Names	Definition	Data type and values	Multiplicity and use
format `format`	A supported output format for the legend image	ows:MimeType	One (mandatory)
minScale Denominator `minScale Denominator`	Minimum scale denominator (inclusive) for which this legend image is valid	Double type, not empty	Zero or one (optional) Include when available and useful [a]
maxScale Denominator `maxScale Denominator`	Maximum scale denominator (exclusive) for which this legend image is valid	Double type, not empty	Zero or one (optional) Include when available and useful [a]
href `href`	The URL from which the legend image can be retrieved	URL type	One (mandatory)

Copyright © 2010 Open Geospatial Consortium Inc.

Names	Definition	Data type and values	Multiplicity and use
width `width`	Width (in pixels) of the legend image	Positive integer type not empty	Zero or one (optional) Include when available and useful
height `height`	Height (in pixels) of the legend image	Positive integer type not empty	Zero or one (optional) Include when available and useful
a The minScaleDenominator and maxScaleDenominator define the range of scales where this legend is valid. The absence of a minScaleDenominator parameter means there is no minimum scale denominator to the condition or logically that the default value is 0. The absence of a maxScaleDenominator parameter means that there is no maximum scale denominator to the condition or logically that the default value is infinity. The absence of both scale parameters in LegendURL metadata means that there is no scale constraint and that the LegendURL is applicable to the style at all scales. General considerations about the meaning of minScaleDenominator and maxScaleDenominator values and their pixel size equivalences as explained in subclause 6.1 also apply here.			

In case of multi-dimensional data, the service metadata can describe their multi-dimensionality and tiles can be requested at specific values in these dimensions. Examples of dimensions are Time, Elevation and Band. Optional parameters in WMTS service metadata declare available values along one or more dimensional axes applicable to a Layer. GetTile and GetFeatureInfo requests for that layer should include parameters specifying dimensional value(s).

Table 9 — Parts of Dimension data structure

Names	Definition	Data type and values	Multiplicity and use
identifier[a] `Identifier`	A name of dimensional axis [e]	ows:CodeType, as adaptation of MD_Identifier class ISO 19115	One (mandatory)
title [b] `Title`	Title of this dimension, normally used for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented [d]
abstract [b] `Abstract`	Brief narrative description of this dimension, normally available for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented
keywords[b] `Keywords`	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this dataset	MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used

Copyright © 2010 Open Geospatial Consortium Inc.

Names	Definition	Data type and values	Multiplicity and use
UOM `UOM`	Units of measure of dimensional axis	DomainMetadata data structure, see Table 43 in OWS Common [OGC 06-121r3]	Zero or one (optional) Include when available and useful
unitSymbol UnitSymbol	Symbol of the units	Character String type, not empty	Zero or one (optional) Include when available and useful
default `Default`	Default value that will be used if a tile request does not specify a value or uses the keyword 'default'	Character String type, not empty (with the exception of 'default' or 'current')	One (mandatory)
current `Current`	A value of 'true' indicates (a) that temporal data are normally kept current and (b) that the request value of this dimension accepts the keyword 'current'	Boolean	Zero or one (optional) Include only for temporal extents. Default is 'false'
value `Value`	Indicates an available value for this dimension	Character String type, not empty	one or more (mandatory)[c] One of each dimension value

a This has the same meaning as "name" in WMS but has been replaced by "identifier" to harmonize with OWS Common.

b The multilingual scoping rules in subclause 10.7.3 of OWS Common [OGC 06-121r3] SHALL apply.

c Repeat this parameter for each available value for this dimension.

d If no Title is specified, client may display the Identifier value instead

e Dimension identifies SHALL be unique (different) for each layer of this server

NOTE 3 The WMS content of a dimension section has a property called multiValues to inform the client that server supports the requesting of multiple values at the same time. The WMTS request for a tile does not support this as a possible property for the dimension data type in this standard.

Table 10 — Parts of TileMatrixSetLink data structure

Names	Definition	Data type and values	Multiplicity and use
tileMatrix Set `Tile Matrix Set`	Reference to a tileMatrixSet	URI type Values SHALL be an tileMatrixSet identifieir in service metadata document	One (mandatory)
tileMatrix Set Limits `Tile Matrix Set Limits`	Index limits for this tileMatrixSet	TileMatrixSetLimits data structure, see Table 11	Zero or one (optional) Should be include when the boundary of the data is a fragment of the boundary of the tileMatrixSet [a]

a The absence of this parameter means that tile row and tile column indices are only limited by 0 and the corresponding matrixWidth and matrixHeight for each tileMatrix of the tileMatrixSet definition.

Copyright © 2010 Open Geospatial Consortium Inc.

Table 11 — Parts of TileMatrixSetLimits data structure

Names	Definition	Data type and values	Multiplicity and use
tileMatrixLimits `TileMatrixLimits`	Indices limits for this tileMatrix	TileMatrixLimits data structure, see Table 12	one or more (mandatory) [a]
a Multiplicity SHALL be the multiplicity of tileMatrix this tileMatrixSet.			

Table 12 — Parts of TileMatrixLimits data structure

Names	Definition	Data type and values	Multiplicity and use
tileMatrix `TileMatrix`	Reference to a tileMatrix identifier	URI type Values defined in service metadata [a]	One (mandatory)
minTileRow `MinTileRow`	Minimum tile row index valid for this layer.	Positive integer type [b]	One (mandatory)
maxTileRow `MaxTileRow`	Maximim tile row index valid for this layer.	Positive integer type [c]	One (mandatory)
minTileCol `MinTileCol`	Minimum tile column index valid for this layer.	Positive integer type [d]	One (mandatory)
maxTileCol `MaxTileCol`	Maximim tile column index valid for this layer.	Positive integer type [e]	One (mandatory)
a URI SHALL be an identifier to a tileMatrix section of this tileMatrixSet for this layer. b From 0 to maxTileRow c From minTileRow to matrixWidth-1 of the tileMatrix section of this tileMatrixSet d From 0 to maxTileCol e From minTileCol to tileHeight-1 of the tileMatrix section of this tileMatrixSet			

A tileMatrixSet defines a generic tiled space bounding box through a TopLeftCorner and MatrixWidth and MatrixHeight as explained in clause 6. For practical reasons some layers that point to this tiled space might not have data covering the entire bounding box but have data covering only some rectangular subset. The optional TileMatrixSetLimits should be included in the description of the layer section to reflect this fact. A request for a tile outside the area marked on Figure 7 SHOULD result in an exception response.

 Copyright © 2010 Open Geospatial Consortium Inc.

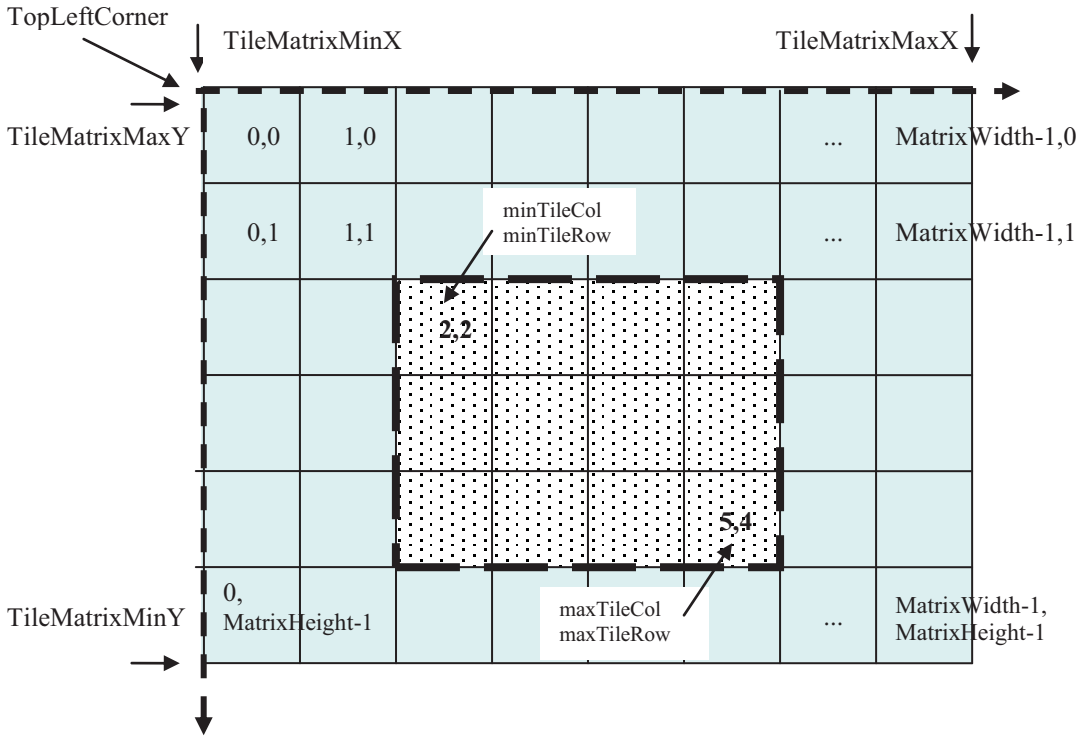


Figure 7 — Optional TileMatrix Limits

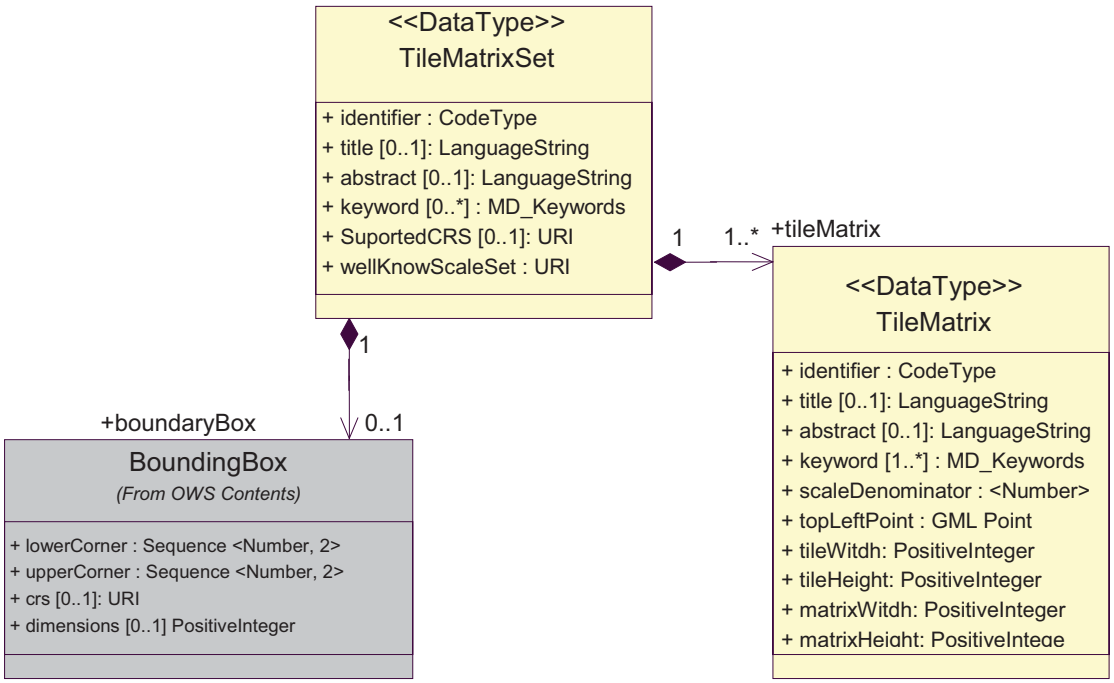


Figure 8 — TileMatrixSet UML model

Copyright © 2010 Open Geospatial Consortium Inc.

Each layer has one or more references to a TileMatrixSet identifier. The structure in Table 13 defines the structure of the TileMatrixSet sections in the Content section.

NOTE 4 If a client requires all tiles to be aligned to a specific TileMatrixSet, it could choose to only display layers that share the same TileMatrixSet identifier. Alternatively, it could compare TileMatrixSet definitions for an equivalency (a simple calculation can be performed to verify whether or not two given tile matrices are aligned).

Table 13 — Parts of TileMatrixSet data structure

Names	Definition	Data type and values	Multiplicity and use
identifier `Identifier`	Tile matrix set identifier [g]	ows:CodeType, as adaptation of MD Identifier class ISO 19115	One (mandatory)
title [a] `Title`	Title of this tile matrix set, normally used for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented [f]
abstract [a] `Abstract`	Brief narrative description of this tile matrix set, normally available for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented
keywords[a] `Keywords`	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this dataset	MD Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
bounding Box `Bounding Box`	Minimum bounding rectangle surrounding the tile matrix set, in the supported CRS [b]	BoundingBox data structure, see subclause 10.2 of OWS Common [OGC 06-121r3]	Zero or one (optional)
supported CRS `Supported CRS`	Reference to one coordinate reference system (CRS)	URI type	One (mandatory)
wellKnown ScaleSet `WellKnown ScaleSet`	Reference to a well known scale set[e]	URI type	Zero or one (optional) [c]
tileMatrix `TileMatrix`	Describes a scale level and its tile matrix	TileMatrix data structure, see Table 14	One or more (mandatory) [d]

 Copyright © 2010 Open Geospatial Consortium Inc.

Names	Definition	Data type and values	Multiplicity and use
a The multilingual scoping rules in subclause 10.7.3 of OWS Common [OGC 06-121r3] SHALL apply.			
b If available, it represents the area where the data is expected to be represented. This does not necessarily indicate a complete tile boundary (and therefore does not necessarily include the TopLeftCorner of the tile matrices).			
c When a tile matrix set conforms to a well-known scale set it SHOULD reference it by its URI. The well-known scale set SHALL be consistent with the supportedCRS and with the scaleDenominators of the tileMatrix parameters.			
d Commonly more than one. Each tileMatrix of a tileMatrixSet SHALL have a unique (different) scaleDenominator			
e Some possible values are defined the in Annex E			
f If no Title is specified, client may display the Identifier value instead			
g TileMatrixSet identifies SHALL be unique (different) for each TileMatrixSet of this server			
h The content of this parameter follows subclause 11.3 and annex D.14 of OWS Common [OGC 06-121r3]			

Table 14 — Parts of TileMatrix data structure

Names	Definition	Data type and values	Multiplicity and use
identifier `Identifier`	Tile matrix identifier [c]	ows:CodeType, as adaptation of MD_Identifier class ISO 19115	One (mandatory)
title [a] `Title`	Title of this style, normally used for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented [d]
abstract [a] `Abstract`	Brief narrative description of this style, normally available for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented
keywords[c] `Keywords`	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this dataset	MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
scale Denominator `Scale Denominator`	Scale denominator level of this tile matrix	Double type	One (mandatory)
topLeftCorner `TopLeftCorner`	Position in CRS coordinates of the top-left corner of this tile matrix	Ordered sequence of double values [b]	One (mandatory)
tileWidth `TileWidth`	Width of each tile of this tile matrix in pixels	Positive integer type	One (mandatory)

Copyright © 2010 Open Geospatial Consortium Inc.

Names	Definition	Data type and values	Multiplicity and use
tileHeight `TileHeight`	Height of each tile of this tile matrix in pixels	Positive integer type	One (mandatory)
matrixWidth `MatrixWidth`	Width of the matrix (number of tiles in width)	Positive integer type	One (mandatory)
matrixHeight `MatrixHeight`	Height of the matrix (number of tiles in height)	Positive integer type	One (mandatory)

a The multilingual scoping rules in subclause 10.7.3 of OWS Common [OGC 06-121r3] SHALL apply.

b CRS will be inherited from the supportedCRS parameter of the parent TileMatrixSet. The order of these axes, shall be as specified by the supportedCRS. These are the precise coordinates of the top left corner of top left pixel of the 0,0 tile. See Figure 2.

c This TileMatrix identifiers SHALL be unique (different) within the context of the parent TileMatrixSet. Consider using a rounded scale denominator or a rounded pixel size as a value.

d If no Title is specified, client may display the Identifier value instead.

e In XML schemas ows:PositionType data type is used. See OWS 1.1 schemas (owsCommon.xsd)

NOTE 5 It may be desirable to define a tile matrix set with some general-scale tile matrices in one CRS (*e.g.*, CRS:84) and with detailed-scale tile matrices in a different CRS (*e.g.*, LCC projection). However, this standard does not allow this. Each tile matrix set SHALL declare a single CRS. You could define two tile matrix sets for the same layer instead.

NOTE 6 The width and height in tiles of each tile matrix is explicitly given, so the range of relevant tile indexes does not have to be calculated by the client application.

NOTE 7 The bounding box of a tile matrix is not supplied explicitly because it can be calculated from topLeftCorner, tileWidth, tileHeight and scaleDenominator.

7.1.1.1.3 *Themes* section contents

The optional Themes section of a WMTS service metadata document SHALL contain data about how layers are organized thematically.

The WMTS standard proposes a different approach from WMS for layer organization, an approach based on the idea of themes. In the Contents section of WMTS, layers are represented as a linear list without hierarchy, and a hierarchy of themes is specified separately in the Themes section, removing the need to specify complex inheritance rules for layer properties. This separates both concepts and makes it easy for a client to ignore the theme hierarchy or even to force another layer organization. Also it allows servers to offer more than one layer organization (in more than one themes section).

Each theme has a human-readable description (*i.e.*, a title) and a list of layer references and child themes. It is possible for a layer to be a member of more than one theme, and for a layer to exist without being a member of any theme.

 Copyright © 2010 Open Geospatial Consortium Inc.

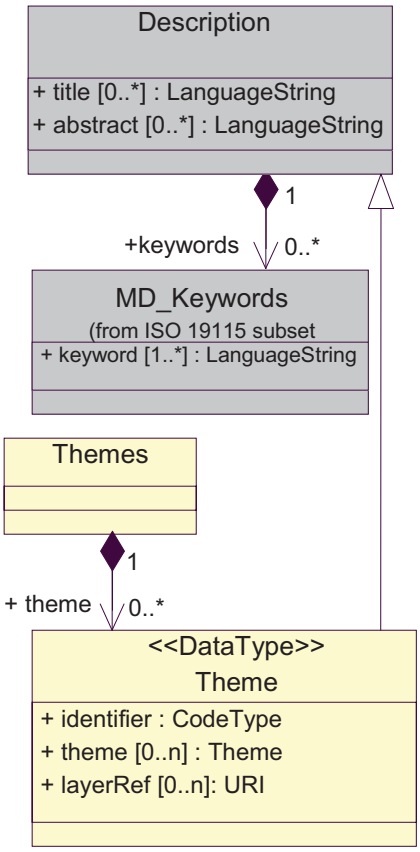


Figure 9 — Themes UML model

The Themes sections SHALL include the parameters specified in Table 15 and Table 16.

Table 15 — Parts of the Themes section

Names	Definition	Data type and values	Multiplicity and use
theme `Theme`	Metadata describing the top-level themes where layers available on this server can be classified	Theme data structure, see Table 16	Zero or more (optional) [a] One for each top-level theme available

Table 16 — Parts of Theme data structure

Names	Definition	Data type and values	Multiplicity and use
identifier `Identifier`	Name of the theme	ows:CodeType, as adaptation of MD_Identifier class ISO 19115	One (mandatory)

Copyright © 2010 Open Geospatial Consortium Inc.

Names	Definition	Data type and values	Multiplicity and use
title [a] `Title`	Title of this theme, normally used for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented [c]
abstract [a] `Abstract`	Brief narrative description of this theme, normally available for display to a human	LanguageString data structure, see Figure 15 in OWS Common [OGC 06-121r3]	Zero or more (optional) Include when available and useful Include one for each language represented
keywords[a] `Keywords`	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this dataset	MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
theme `Theme`	Metadata describing the child (subordinate) themes of this theme where layers available on this server can be classified	Theme data structure, see this Table	Zero or more (optional) [a] One for each theme available
layerRef `LayerRef`	Reference to layer	URI type Values defined in service metadata [b]	Zero or more (optional)

a The multilingual scoping rules in subclause 10.7.3 of OWS Common [OGC 06-121r3] SHALL apply.

b A layer identifier on this ServiceMetadata document.

c If no Title is specified, client may display the Identifier value instead.

7.1.1.2 ServiceMetadata document XML schema

ServiceMetadata documents can be encoded in XML. This standard provides XML schemas to encode XML Service metadata documents as described in Annex B. The following XML schema fragment for a WMTS service metadata document shows how WMTS extends ows:CapabilitiesBaseType in owsCommon.xsd of OWS Common [OGC 06-121r3] to include other parameters as previously described in Table 3:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wmts="http://www.opengis.net/wmts/1.0"
        xmlns:ows="http://www.opengis.net/ows/1.1"
        xmlns:gml="http://www.opengis.net/gml"
        targetNamespace="http://www.opengis.net/wmts/1.0"
        elementFormDefault="qualified" xml:lang="en">
    <annotation>
        <documentation>
            This XML Schema Document defines the ServiceMetadata document
            namespace.
```


Copyright © 2010 Open Geospatial Consortium Inc.

```
        </documentation>
    </annotation>
    <!-- ==========================================================
        includes and imports
    ========================================================== -->
    <import namespace="http://www.opengis.net/ows/1.1"
          schemaLocation="../../ows/1.1.0/owsAll.xsd"/>
    <!--==========================================================
        elements and types
    ========================================================== -->
    <element name="Capabilities">
        <complexType>
            <complexContent>
                <extension base="ows:CapabilitiesBaseType">
                    <sequence>
                        <element name="Contents" type="wmts:ContentsType"
                              minOccurs="0" />
                        <element ref="wmts:Themes" minOccurs="0"
                              maxOccurs="unbounded" />
                        <element name="WSDL" type="ows:OnlineResourceType"
                              minOccurs="0" maxOccurs="unbounded" />
                        <element name="ServiceMetadataURL"
                              type="ows:OnlineResourceType" minOccurs="0"
                              maxOccurs="unbounded" />
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>
...
</schema>
```

As indicated above, this XML Schema document uses the owsServiceIdentification.xsd, owsServiceProvider.xsd, and owsOperationsMetadata.xsd schemas specified in OWS Common [OGC 06-121r3]. It also uses an XML Schema document for the “Contents” sections of the WMTS ServiceMetadata XML document, which is included in the wmtsGetCapabilities_response.xsd file. All these XML Schema documents contain documentation of the meaning of each element, attribute and type, and this documentation SHALL be considered normative as specified in subclause 11.6.3 of OWS Common [OGC 06-121r3].

Annex B contains more details on this normative set of XML Schema documents.

7.1.1.3 ServiceMetadata document example

A WMTS server might generate a ServiceMetadata document that looks like the following example. Another example can be found in Annex D:

```
<?xml version="1.0" encoding="UTF-8"?>
<Capabilities xmlns="http://www.opengis.net/wmts/1.0"
          xmlns:ows="http://www.opengis.net/ows/1.1"
          xmlns:xlink="http://www.w3.org/1999/xlink"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:gml="http://www.opengis.net/gml"
          xsi:schemaLocation="http://www.opengis.net/wmts/1.0
```

Copyright © 2010 Open Geospatial Consortium Inc.

```
                ../wmtsGetCapabilities_response.xsd"
      version="1.0.0">
  <ows:ServiceIdentification>
    <ows:Title>World example Web Map Tile Service</ows:Title>
    <ows:Abstract>
      Example service that constrains some world layers in the
      GlobalCRS84Pixel Well-known scale set
    </ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>World</ows:Keyword>
      <ows:Keyword>Global</ows:Keyword>
      <ows:Keyword>Digital Elevation Model</ows:Keyword>
      <ows:Keyword>Administrative Boundaries</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>OGC WMTS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
    <ows:Fees>none</ows:Fees>
    <ows:AccessConstraints>none</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>UAB-CREAF-MiraMon</ows:ProviderName>
    <ows:ProviderSite xlink:href="http://www.creaf.uab.cat/miramon"/>
    <ows:ServiceContact>
      <ows:IndividualName>Joan Maso Pau</ows:IndividualName>
      <ows:PositionName>Senior Software Engineer</ows:PositionName>
      <ows:ContactInfo>
        <ows:Phone>
          <ows:Voice>+34 93 581 1312</ows:Voice>
          <ows:Facsimile>+34 93 581 4151</ows:Facsimile>
        </ows:Phone>
        <ows:Address>
          <ows:DeliveryPoint>Fac Ciencies UAB</ows:DeliveryPoint>
          <ows:City>Bellaterra</ows:City>
          <ows:AdministrativeArea>Barcelona
            </ows:AdministrativeArea>
          <ows:PostalCode>08193</ows:PostalCode>
          <ows:Country>Spain</ows:Country>
          <ows:ElectronicMailAddress>joan.maso@uab.cat
            </ows:ElectronicMailAddress>
        </ows:Address>
      </ows:ContactInfo>
    </ows:ServiceContact>
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:href="http://www.maps.bob/maps.cgi?">
            <ows:Constraint name="GetEncoding">
              <ows:AllowedValues>
                <ows:Value>KVP</ows:Value>
              </ows:AllowedValues>
            </ows:Constraint>
          </ows:Get>
          <ows:Post xlink:href="http://www.maps.bob/maps.cgi?">
            <ows:Constraint name="PostEncoding">
              <ows:AllowedValues>
                <ows:Value>SOAP</ows:Value>
```

 Copyright © 2010 Open Geospatial Consortium Inc.

```
                    </ows:AllowedValues>
                </ows:Constraint>
            </ows:Post>
        </ows:HTTP>
    </ows:DCP>
</ows:Operation>
<ows:Operation name="GetTile">
    <ows:DCP>
        <ows:HTTP>
            <ows:Get xlink:href="http://www.maps.bob/maps.cgi?">
                <ows:Constraint name="GetEncoding">
                    <ows:AllowedValues>
                        <ows:Value>KVP</ows:Value>
                    </ows:AllowedValues>
                </ows:Constraint>
            </ows:Get>
        </ows:HTTP>
    </ows:DCP>
</ows:Operation>
<ows:Operation name="GetFeatureInfo">
    <ows:DCP>
        <ows:HTTP>
            <ows:Get xlink:href="http://www.maps.bob/maps.cgi?">
                <ows:Constraint name="GetEncoding">
                    <ows:AllowedValues>
                        <ows:Value>KVP</ows:Value>
                    </ows:AllowedValues>
                </ows:Constraint>
            </ows:Get>
        </ows:HTTP>
    </ows:DCP>
</ows:Operation>
</ows:OperationsMetadata>
<Contents>
    <Layer>
        <ows:Title>etopo2</ows:Title>
        <ows:Abstract>
            ETOPO2 - 2 minute Worldwide Bathymetry/Topography
            Data taken from National Geophysical Data Center(NGDC),
            ETOPO2 Global 2' Elevations, September 2001...
        </ows:Abstract>
        <ows:WGS84BoundingBox>
            <ows:LowerCorner>-180 -90</ows:LowerCorner>
            <ows:UpperCorner>180 90</ows:UpperCorner>
        </ows:WGS84BoundingBox>
        <ows:Identifier>etopo2</ows:Identifier>
        <ows:Metadata
              xlink:href="http://www.maps.bob/etopo2/ metadata.htm"/>
        <Style isDefault="true">
            <ows:Title>default</ows:Title>
            <ows:Identifier>default</ows:Identifier>
            <LegendURL format="image/png"
              xlink:href="http://www.maps.bob/etopo2/legend.png" />
        </Style>
        <Format>image/png</Format>
        <InfoFormat>application/gml+xml; version=3.1</InfoFormat>
        <TileMatrixSetLink>
            <TileMatrixSet>WholeWorld_CRS_84</TileMatrixSet>
```

Copyright © 2010 Open Geospatial Consortium Inc.

```
        </TileMatrixSetLink>
    </Layer>
    <Layer>
        <ows:Title>Administrative Boundaries</ows:Title>
        <ows:Abstract>The sub Country Administrative Units 1998
              GeoDataset represents a small-scale political map of
              the world...</ows:Abstract>
        <ows:WGS84BoundingBox>
            <ows:LowerCorner>-180 -90</ows:LowerCorner>
            <ows:UpperCorner>180 84</ows:UpperCorner>
        </ows:WGS84BoundingBox>
        <ows:Identifier>AdminBoundaries</ows:Identifier>
        <ows:Metadata
    xlink:href="http://www.maps.bob/AdminBoundaries/metadata.htm"/>
        <Style isDefault="true">
            <ows:Title>default</ows:Title>
            <ows:Identifier>default</ows:Identifier>
        </Style>
        <Format>image/png</Format>
        <TileMatrixSetLink>
            <TileMatrixSet>WholeWorld_CRS_84</TileMatrixSet>
        </TileMatrixSetLink>
    </Layer>
    <TileMatrixSet>
        <ows:Identifier>WholeWorld_CRS_84</ows:Identifier>
        <ows:SupportedCRS>urn:ogc:def:crs:OGC:1.3:CRS84
            </ows:SupportedCRS>
        <WellKnownScaleSet>urn:ogc:def:wkss:OGC:1.0:GlobalCRS84Pixel
            </WellKnownScaleSet>
        <TileMatrix>
            <ows:Identifier>2g</ows:Identifier>
            <ScaleDenominator>795139219.9519541</ScaleDenominator>
            <!-- top left point of tile matrix bounding box -->
            <TopLeftCorner>-180 90</TopLeftCorner>
            <!-- width and height of each tile in pixel units -->
            <TileWidth>320</TileWidth>
            <TileHeight>200</TileHeight>
            <!-- width and height of matrix in tile units -->
            <MatrixWidth>1</MatrixWidth>
            <MatrixHeight>1</MatrixHeight>
        </TileMatrix>
        <TileMatrix>
            <ows:Identifier>1g</ows:Identifier>
            <ScaleDenominator>397569609.9759771</ScaleDenominator>
            <TopLeftCorner>-180 90</TopLeftCorner>
            <TileWidth>320</TileWidth>
            <TileHeight>200</TileHeight>
            <MatrixWidth>2</MatrixWidth>
            <MatrixHeight>1</MatrixHeight>
        </TileMatrix>
        <TileMatrix>
            <ows:Identifier>30m</ows:Identifier>
            <ScaleDenominator>198784804.9879885</ScaleDenominator>
            <TopLeftCorner>-180 90</TopLeftCorner>
            <TileWidth>320</TileWidth>
            <TileHeight>200</TileHeight>
            <MatrixWidth>3</MatrixWidth>
            <MatrixHeight>2</MatrixHeight>
```


Copyright © 2010 Open Geospatial Consortium Inc.

```
            </TileMatrix>
            <TileMatrix>
               <ows:Identifier>20m</ows:Identifier>
               <ScaleDenominator>132523203.3253257</ScaleDenominator>
               <TopLeftCorner>-180 90</TopLeftCorner>
               <TileWidth>320</TileWidth>
               <TileHeight>200</TileHeight>
               <MatrixWidth>4</MatrixWidth>
               <MatrixHeight>3</MatrixHeight>
            </TileMatrix>
            <TileMatrix>
               <ows:Identifier>10m</ows:Identifier>
               <ScaleDenominator>66261601.66266284</ScaleDenominator>
               <TopLeftCorner>-180 90</TopLeftCorner>
               <TileWidth>320</TileWidth>
               <TileHeight>200</TileHeight>
               <MatrixWidth>7</MatrixWidth>
               <MatrixHeight>6</MatrixHeight>
            </TileMatrix>
            <TileMatrix>
               <ows:Identifier>5m</ows:Identifier>
               <ScaleDenominator>33130800.83133142</ScaleDenominator>
               <TopLeftCorner>-180 90</TopLeftCorner>
               <TileWidth>320</TileWidth>
               <TileHeight>200</TileHeight>
               <MatrixWidth>14</MatrixWidth>
               <MatrixHeight>11</MatrixHeight>
            </TileMatrix>
            <TileMatrix>
               <ows:Identifier>2m</ows:Identifier>
               <ScaleDenominator>13252320.33253257</ScaleDenominator>
               <TopLeftCorner>-180 84</TopLeftCorner>
               <TileWidth>320</TileWidth>
               <TileHeight>200</TileHeight>
               <MatrixWidth>34</MatrixWidth>
               <MatrixHeight>28</MatrixHeight>
            </TileMatrix>
         </TileMatrixSet>
      </Contents>
      <Themes>
         <Theme>
            <ows:Title>Foundation</ows:Title>
            <ows:Abstract>World reference data</ows:Abstract>
            <ows:Identifier>Foundation</ows:Identifier>
            <Theme>
               <ows:Title>Digital Elevation Model</ows:Title>
               <ows:Identifier>DEM</ows:Identifier>
               <LayerRef>etopo2</LayerRef>
            </Theme>
            <Theme>
               <ows:Title>Administrative Boundaries</ows:Title>
               <ows:Identifier>AdmBoundaries</ows:Identifier>
               <LayerRef>AdminBoundaries</LayerRef>
            </Theme>
         </Theme>
      </Themes>
</Capabilities>
```

Copyright © 2010 Open Geospatial Consortium Inc.

7.1.2 GetCapabilities operation (mandatory in procedure oriented architectural style)

The GetCapabilities operation in procedure oriented architectural style allows WMTS clients to retrieve a service metadata document from a server. The response to a GetCapabilities request SHALL be a document containing service metadata about the server, including specific information about the layers that can be requested, the tile sets in which these layers are available, and (optionally) one or more theme sets. The GetCapabilities operation also includes a version-negotiation mechanism, allowing the client and server to agree on a standard version on which to base all future communication. The subclause 7.1.1 specifies the sections for the ServiceMetadata document that a WMTS server SHALL return to describe its service metadata (usually encoded in an XML file).

7.1.2.1 GetCapabilities operation request

The GetCapabilities operation request SHALL be as specified in Subclauses 7.2 and 7.3 of OWS Common [OGC 06-121r3] and SHALL follow the Tables 3 and 7 of OWS Common [OGC 06-121r3]. Parameters of the GetCapabilities request are described in Table 3 of OWS Common [OGC 06-121r3] with the restriction in the value of the "service" parameter that SHALL be "WMTS" as listed in Table 17 below.

Table 17 — Parameters in GetCapabilities operation request

Names	Definition	Data type and values	Multiplicity and use
service `Service`	Service type identifier	Character String type, not empty SHALL be "WMTS"	One (mandatory)
request `Request`	Operation name	Character String type, not empty SHALL be "GetCapabilities"	One (mandatory)
accept Versions `Accept Versions`	Prioritized sequence of one or more standard versions accepted by client, with preferred versions listed first	Sequence of Character String type, each not empty Value is list of x.y.z "version" values. SHALL contain "1.0.0"	Zero or one (optional) When omitted, return latest supported version
sections `Sections`	Unordered list of zero or more names of requested sections in complete service metadata document	Sequence of Character String type, each not empty Value is list of section names Allowed section names are in Table 18	Zero or one (optional) When omitted or not supported by server, return complete service metadata document
update Sequence `Update Sequence`	Service metadata document version, value is "increased" whenever any change is made in complete service metadata document	Character String type, not empty Values are selected by each server, and are always opaque to clients	Zero or one (optional) When omitted or not supported by server, return latest service metadata document

 Copyright © 2010 Open Geospatial Consortium Inc.

Names	Definition	Data type and values	Multiplicity and use
accept Formats `Accept Formats`	Prioritized sequence of zero or more response formats desired by client, with preferred formats listed first	Sequence of Character String type, each not empty Value is list of format identifiers Identifiers are MIME types of formats useful for service metadata documents	Zero or one (optional) When omitted or not supported by server, return service metadata document using MIME type "application/xml"

Sections name values of the WMTS service valid in a GetCapabilities request are specified in Table 6 of OWS Common [OGC 06-121r3] with the addition of the Themes section as listed in Table 18 below.

Table 18 — Meaning of section name values

Section name	Meaning
ServiceIdentification	Return ServiceIdentification element in service metadata document
ServiceProvider	Return ServiceProvider metadata element in service metadata document
OperationsMetadata	Return OperationsMetadata element in service metadata document
Contents	Return Contents metadata element in service metadata document
Themes	Return Themes metadata element in service metadata document
All	Return complete service metadata document, containing all elements

The “Multiplicity and use” column in Table 1 of OWS Common [OGC 06-121r3] specifies the optionally of each listed parameter in the GetCapabilities operation request. Table 19 specifies the implementation of those parameters by WMTS clients and servers.

Table 19 — Implementation of parameters in GetCapabilities operation request

Names	Multiplicity	Client implementation	Server implementation
service `Service`	One (mandatory)	Each parameter SHALL be implemented by all clients, using specified value	Each parameter SHALL be implemented by all servers, checking that each parameter is received with specified value
request `Request`	One (mandatory)		
acceptVersions `Accept Versions`	Zero or one (optional)	SHOULD be implemented by all software clients, using specified values	SHALL be implemented by all servers, checking if parameter is received with specified value(s)
sections `Sections`	Zero or one (optional)	Each parameter may be implemented by each client If parameter not provided, SHALL expect default response	Each parameter may be implemented by each server If parameter not implemented or not received, SHALL provide default response
updateSequence `Update Sequence`	Zero or one (optional)		

Copyright © 2010 Open Geospatial Consortium Inc.

acceptFormats `AcceptFormats`	Zero or one (optional)	If parameter provided, SHALL allow default or specified response	If parameter implemented and received, SHALL provide specified response

7.1.2.2 GetCapabilites exceptions in procedure oriented architectural style

When a WMTS server encounters an error while performing a GetCapabilities operation, it SHALL return an exception-report message as specified in clause 8 of OWS Common [OGC 06-121r3]. The allowed exception codes SHALL include those listed in Table 20 assuming the updateSequence parameter is implemented by the server.

NOTE 1 To reduce the need for readers to refer to other documents, the table below is copied from Table 8 in subclause 7.4.1 of OWS Common [OGC 06-121r3].

Table 20 — Exception codes for GetCapabilities operation

exceptionCode value	Meaning of code	"locator" value
MissingParameterValue	Operation request does not include a parameter value	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
VersionNegotiationFailed	List of versions in "AcceptVersions" parameter value, in GetCapabilities operation request, did not include any version supported by this server	None, omit "locator" parameter
InvalidUpdateSequence	Value of (optional) updateSequence parameter, in GetCapabilities operation request, is greater than current value of service metadata updateSequence number	None, omit "locator" parameter
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

If the client sends a KVP encoded request using unknown parameters these unknown parameters SHALL be ignored by the server and will not cause an exception to be generated.

 Copyright © 2010 Open Geospatial Consortium Inc.

When a WMTS server responds with an ExceptionReport and the report is transmitted via HTTP, the WMTS server should set the status code of the HTTP response to the corresponding value for the given exceptionCode values, as shown in Table 21. When the ExceptionReport contains more than one Exception, then the HTTP status code value should be based upon the exceptionCode of the first Exception in the ExceptionReport.

Table 21 — HTTP exception codes and meanings on GetCapabilities operation

	HTTP Status Code	
exceptionCode value	**Code**	**Message**
MissingParameterValue	400	Bad request
InvalidParameterValue	400	Bad request
VersionNegotiationFailed	400	Bad request
InvalidUpdateSequence	400	Bad request
NoApplicableCode	500	Internal server error

7.1.3 ServiceMetadata resource request (mandatory in resource oriented architectural style)

WMTS servers using a resource oriented architectural style provide standard endpoints from which a representation of the ServiceMetadata resource can be obtained. The endpoint SHALL also be specified in the ServiceMetadata document although it will generally be obtained prior to communication with the server.

The client will request the representation of the ServiceMetadata resource by performing a standard request to the endpoint. In response to a correct request, the server SHALL provide a representation of its ServiceMetadata document which conforms with subclause 7.1.1. Incorrect requests shall be handled according to the standard semantics for errors for the transport protocol used for communication between the client and the server.

7.2 Tile

WMTS servers are designed to serve map image tiles. The ServiceMetadata document described in the previous subclause lists the tiles available on the server and the requirements for requesting a tile. Typically clients will first request the ServiceMetadata document from the server and then use the information in that document to discover how to perform valid requests for tiles.

Copyright © 2010 Open Geospatial Consortium Inc.

7.2.1 Tile resource

The tile resource is generally a rectangular image containing cartographic data. Alternatively, this resource might be a non-image representation of the tile such as a description of the tile or a link to the actual image. For example, the tile resource could be a KML document used in a superoverlay, or a tile metadata document. When returning an image tile, a full single tile SHALL always be returned. Also, the background pixels of a tile SHOULD be transparent when possible so that the client can overlay the tiles on top of other map data (possibly other tiles).

The Tile resource representation SHALL be returned in the format specified in the request when the format has been advertised in the ServiceMetadata document as available for that Tile resource.

7.2.2 GetTile operation (mandatory in procedure oriented architectural style)

The GetTile operation in procedure oriented architectural style allows WMTS clients to request a particular tile of a particular tile matrix set in a predefined format. This operation has some parameters in common with WMS GetMap but it has been deliberately simplified. For instance, only one layer can be retrieved at a time. WMTS servers that want to allow a combination of layers to be served and requested have to give this combination an identifier and add it as a new layer in the service metadata document. Nevertheless, clients are expected to be able easily to overlay layers themselves eliminating the need for servers to offer layers by combination.

7.2.2.1 GetTile operation request

A request to perform the GetTile operation SHALL use the data structure specified in Table 22. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the default server behavior when an optional parameter is not included in the operation request.

 Copyright © 2010 Open Geospatial Consortium Inc.

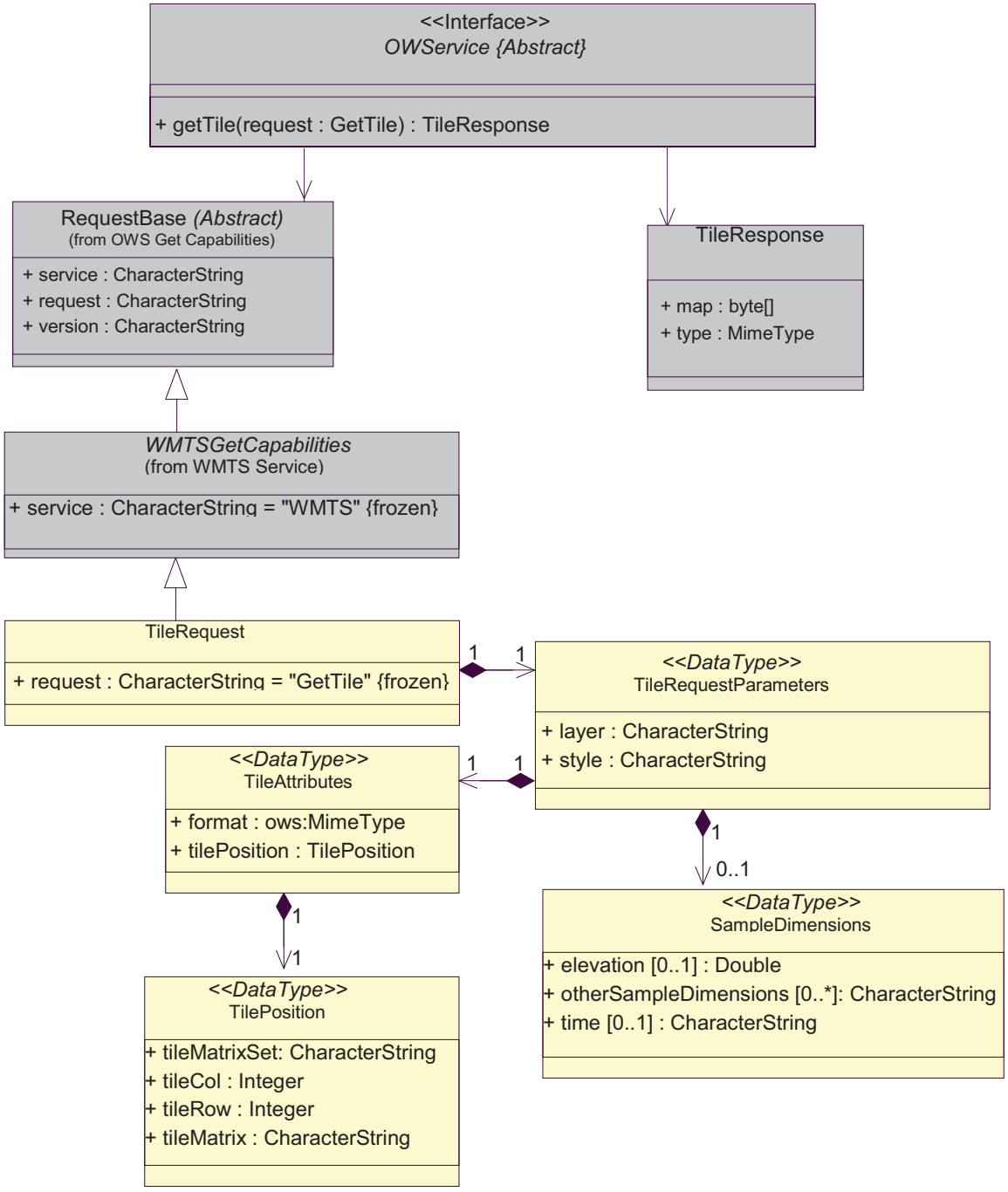


Figure 10 — GetTile operation request UML class diagram

Copyright © 2010 Open Geospatial Consortium Inc.

Table 22 — Parameters in GetTile operation request

Names	Definition	Data type and values	Multiplicity and use
service `Service`	Service type identifier	Character String type, not empty SHALL be "WMTS"	One (mandatory)
request `Request`	Operation name	Character String type, not empty SHALL be "GetTile"	One (mandatory)
version `Version`	Standard version for operation	Character String type, not empty SHALL contain "1.0.0"	One (mandatory)
layer `Layer`	Layer identifier	Character String type, not empty identifier that is defined in the ServiceMetadata document	One (mandatory)
style `Style`	Style identifier	Character String type, not empty identifier that is defined in the ServiceMetadata document	One (mandatory)
format `Format`	Output format of the tile	Character String type, not empty value that is defined in the ServiceMetadata document	One (mandatory)
Other sample dimensions [a]	Value allowed for this dimension	Character String type, not empty a single value from a list or a range defined in the ServiceMetadata document	Zero or one (optional)
tileMatrix Set `TileMatrix Set`	TileMatrixSet identifier	Character String type, not empty identifier that is defined in the ServiceMetadata document	One (mandatory)
tileMatrix `TileMatrix`	TileMatrix identifier [b]	Character String type, not empty value that is defined in the ServiceMetadata document	One (mandatory)
tileRow `TileRow`	Row index of tile matrix	Non negative integer type value between 0 and MatrixHeight-1 of this tile matrix defined in the ServiceMetadata document	One (mandatory)
tileCol `TileCol`	Column index of tile matrix	Non negative integer type value between 0 and MatrixWidth-1 of this tile matrix defined in the ServiceMetadata document	One (mandatory)

a The name of this parameter is, in fact, the identifier of the dimension specified in the ServiceMetadata. See WMS 1.3.0 annex C for reference. This parameter appear once for each dimension specified for this Layer in the ServiceMetadata document.

b This will be the identifier of the tileMatrix of the desired scale denominator for the tileMatrixSet requested.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in subclause 9.2.1 of OWS Common [OGC 06-121r3].

NOTE 2 The UML class diagram in Figure 10 provides a useful graphical view of the contents of the GetTile operation request listed in Table 22.

Copyright © 2010 Open Geospatial Consortium Inc.

7.2.2.2 GetTile exceptions in procedure oriented architectural style

When a WMTS server encounters an error while performing a GetTile operation, it SHALL return an exception report message as specified in subclause 7.4.1 of OWS Common [OGC 06-121r3]. The allowed standard exception codes SHALL include those listed in Table 23. For each listed exceptionCode, the contents of the "locator" parameter value SHALL be as specified in the right column of Table 23.

NOTE 1 To reduce the need for readers to refer to other documents, four values listed below are copied from Table 8 in subclause 7.4.1 of OWS Common [OGC 06-121r3].

Table 23 — Exception codes for GetTile operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
TileOutOfRange	TileRow or TileCol out of range	Name of the out-of-range parameter
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

If the client sends a GetTile request using unknown parameters (for example time, elevation or any other dimension that are not advertised in the ServiceMetadata document) these unknown parameters SHALL be ignored by the server and will not cause an exception to be generated.

When a WMTS server responds with an ExceptionReport and the report is transmitted via HTTP, the WMTS server should set the status code of the HTTP response to the corresponding value for the given exceptionCode values, as shown in Table 24. When the ExceptionReport contains more than one Exception then the HTTP status code value should be based upon the exceptionCode of the first Exception in the ExceptionReport.

Table 24 — HTTP exception codes and meanings on GetTile operation

	HTTP Status Code	
exceptionCode value	Code	Message
OperationNotSupported	501	Not implemented
MissingParameterValue	400	Bad request
InvalidParameterValue	400	Bad request
TileOutOfRange	400	Bad request
NoApplicableCode	500	Internal server error

Copyright © 2010 Open Geospatial Consortium Inc.

7.2.3 Tile resource request (mandatory in resource oriented architectural style)

WMTS servers using a resource oriented architectural style provide standard endpoints from which a representation of each Tile resource can be obtained. The endpoints SHALL be specified in the ServiceMetadata document using an address template.

The client will request the representation of any offered Tile resource by performing a request to the address following the standard semantics of the transport protocol used for communication between the client and the server. In response to a correct request, the server SHALL provide a representation of the Tile resource. Incorrect requests shall be handled according to the standard semantics for the transport protocol.

7.3 FeatureInfo

WMTS servers may support requests for information about the features present at a particular pixel location on a map tile. Requests for feature information will specify the tile along with a pixel location on that tile. The WMTS server will provide information on the features present at or near the location specified by the client request. The WMTS server may choose what information to provide about the nearby features.

7.3.1 FeatureInfo document

A FeatureInfo document is either the response of a GetFeatureInfo request in procedure oriented architectural style or the resource representation of a FeatureInfo resource in resource oriented architectural style. The FeatureInfo document SHALL be in the format specified in the request when that format has been advertised in the ServiceMetadata document as available for that FeatureInfo resource.

For a better interoperability between servers and clients we strongly recommend GML Simple Features Profile [06-049r1] as a supported document format for FeatureInfo resources. That standard defines three levels of content in three profiles with different degrees of constraints to the GML flexibility. We strongly recommend support of the most constrained one (level 0) that results in a simpler GML document. In the context of that profile only simple XML types can be used as thematic properties and cardinality greater than one is not allowed. Servers and clients SHALL specify the MIME type "application/gml+xml; version=3.1" as an InfoFormat value and the GML application schema of the response SHOULD conform to GML Simple Features profile level 0 when that GML profile is used. In most cases, only thematic attributes of the features are intended to be included in a FeatureInfo document but the Simple Feature profiles were evidently intended to include the geometric information of the features in the GML objects. However, it is possible to generate an application schema that does not include feature geometry and only describes non-geometric feature attribute types. This can be very useful to avoid unnecessarily requesting long sequences of position values in line or polygon layers.

 Copyright © 2010 Open Geospatial Consortium Inc.

Also, to allow easy presentation of the data, support for the HTML format (represented by an InfoFormat MIME type of “text/html”) is also recommended.

NOTE OGC 06-049r1 recommends the use of "text/xml; subType=gml/3.1.1/profiles/gmlsf/1.0.0/0" but this has been corrected by OGC 09-144r1 Technical Committee Policies and Procedures: MIME Media Types for GML. This document adopts the new policy.

7.3.2 GetFeatureInfo operation (optional in procedure oriented architectural style)

The GetFeatureInfo operation in procedure oriented architectural style allows WMTS clients to request information at a particular position of a particular tile for a particular queryable layer. A layer is queryable if the Contents section of the ServiceMetadata document specifies one or more InfoFormats for this layer.

NOTE 1 This criterion is different from the one used in WMS. In WMTS, the queryable property of WMS has been substituted by the presence or absence of an InfoFormat element in the ServiceMetadata document.

The GetFeatureInfo operation is designed to provide clients of a WMTS with more information about features rendered in a previously returned tile. The canonical use case for GetFeatureInfo is that a user chooses a pixel (I,J) on a particular tile at which the user would like to obtain more information. Because the WMTS protocol is stateless, the GetFeatureInfo request indicates to the WMTS server what tile the user is viewing by including the original GetTile request parameters but modifying the request value to 'GetFeatureInfo' and adding the pixel offset parameters. From the spatial context information (TileRow, TileCol and TileMatrixSet), along with the I,J position the user requested, the WMTS can return additional information about that position. The other GetTile parameters (*e.g.*, Style) may play a role in the server's decision as to what information to return.

NOTE 2 When the user chooses a point (I,J) in a client that is showing overlapped layers, the client will need to make a separate GetFeatureInfo request for each layer.

7.3.2.1 GetFeatureInfo operation request

A request to perform the GetFeatureInfo operation SHALL include the use of the data parameters as specified in Figure 11 and in Table 25. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Copyright © 2010 Open Geospatial Consortium Inc.

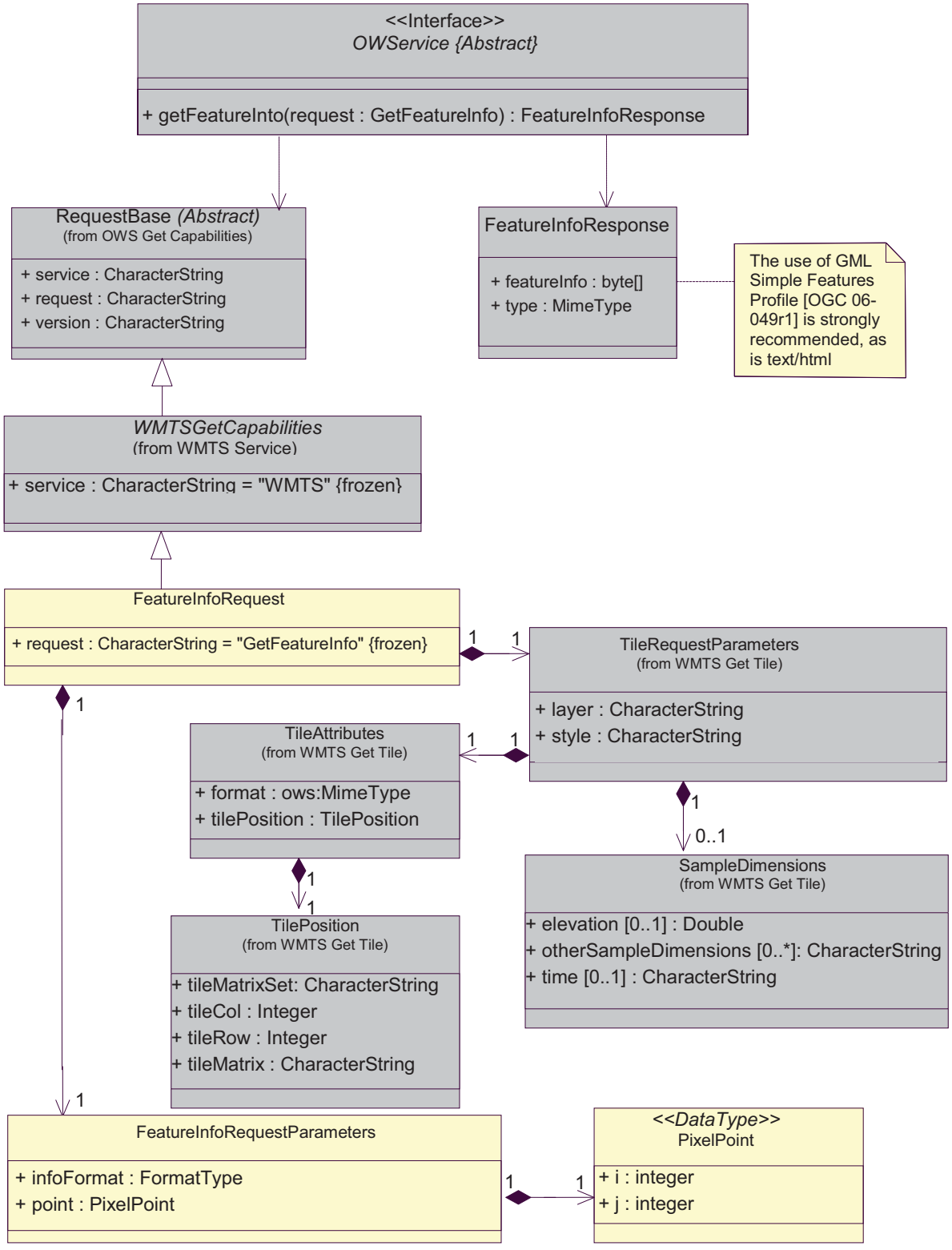


Figure 11 — GetFeatureInfo operation request UML class diagram

Copyright © 2010 Open Geospatial Consortium Inc.

Table 25 — Parameters in GetFeatureInfo operation request

Names	Definition	Data type and values	Multiplicity and use
service `Service`	Service type identifier	Character String type, not empty SHALL be “WMTS”	One (mandatory)
request `Request`	Operation name	Character String type, not empty SHALL be "GetFeatureInfo”	One (mandatory)
version `Version`	Standard version for operation	Character String type, not empty SHALL contain 1.0.0	One (mandatory)
layer, style, format, *Sample dimension,* tileMatrixSet, tileMatrix, tileRow, tileCol	These correspond to the parameters of the same name in the corresponding GetTile request described in Table 22	The values of these parameters SHALL match those in the corresponding GetTile request described in Table 22	Multiplicity and use of these parameters SHALL match those in the corresponding GetTile request described in Table 22
j `J`	Row index of a pixel within the tile[a]	Non negative integer type value between 0 and TileHeight-1 of this tile matrix defined in the ServiceMetadata document	One (mandatory)
i `I`	Column index of a pixel within the tile[b]	Non negative integer type value between 0 and TileWidth-1 of this tile matrix defined in the ServiceMetadata document	One (mandatory)
infoFormat `InfoFormat`	Output format of the retrieved information	Character String type, not empty value that is defined in the ServiceMetadata document	One (mandatory)
a Number of full pixels in the Tile to the left of the requested location b Number of full pixels in the Tile to the top of the requested location			

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed are largely copied from Table 26 in subclause 9.2.1 of OWS Common [OGC 06-121r3].

NOTE 2 The UML class diagram in Figure 11 provides a useful graphical view of the contents of the GetFeatureInfo operation request listed in Table 25.

Although some values listed in the "Names" column appear to contain white spaces, they SHALL not contain any spaces.

7.3.2.2 GetFeatureInfo exceptions in procedure oriented architectural style

When a WMTS server encounters an error while performing a GetFeatureInfo operation, it SHALL return an exception report message as specified in subclause 7.4 of OWS Common [OGC 06-121r3]. The allowed standard exception codes SHALL include those listed in Table 26. For each listed exceptionCode, the contents of the “locator” parameter value SHALL be as specified in the right column of Table 26.

Copyright © 2010 Open Geospatial Consortium Inc.

Table 26 — Exception codes for GetFeatureInfo operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server or you are requesting a GetFeatureInfo response for a non queryable layer	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
TileOutOfRange	TileRow or TileCol out of range	Name of the out-of-range parameter
PointIJOutOfRange	I or J out of range	Name of the out-of-range parameter
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 25 in subclause 8.3 of OWS Common [OGC 06-121r3].

If the client sends a GetFeatureInfo request using unknown parameters (for example time, elevation or any other dimension that are not advertised in the ServiceMetadata document) these unknown parameters SHALL be ignored by the server and will not cause an exception to be generated.

When a WMTS server responds with an ExceptionReport and the report is transmitted via HTTP the WMTS server should set the HTTP response's status code to the corresponding value for the given exceptionCode values, as shown in Table 27. When the ExceptionReport contains more than one Exception then the HTTP status code value should be based upon the exceptionCode of the first Exception in the ExceptionReport.

Table 27 — HTTP exception codes and meanings on GetFeatureInfo operation

	HTTP Status Code	
exceptionCode value	Code	Message
OperationNotSupported	501	Not implemented
MissingParameterValue	400	Bad request
InvalidParameterValue	400	Bad request
TileOutOfRange	400	Bad request
PointIJOutOfRange	400	Bad request
NoApplicableCode	500	Internal server error

Copyright © 2010 Open Geospatial Consortium Inc.

7.3.3 FeatureInfo resource request (optional in resource oriented architectural style)

WMTS servers using a resource oriented architectural style provide standard endpoints from which representations of the FeatureInfo resources can be obtained. The endpoints SHALL be specified in the ServiceMetadata document based on an address template.

The client will request the representation of a FeatureInfo resource by performing a request to the address following the standard semantics of the transport protocol used for communication between the client and the server. In response to a correct request, the server SHALL provide a representation of each FeatureInfo resource. Incorrect requests shall be handled according to the standard semantics for the transport protocol.

7.4 Operation request encoding

The encoding of procedure oriented architectural style operation requests performed over HTTP SHALL use HTTP GET with KVP encoding or HTTP POST with KVP or SOAP encoding as specified in clause 11 of OWS Common [OGC 06-121r3]. Table 28 summarizes the WMTS Service operations and their encoding methods defined in this standard.

Table 28 — Procedure oriented architectural style operation request encoding

Operation name	Request encodings
GetCapabilities (required)	KVP, SOAP
GetTile (required)	KVP, SOAP
GetFeatureInfo	KVP, SOAP

HTTP GET with KVP is described in clause 8 and HTTP POST with SOAP is defined in described in clause 9.

A resource oriented architectural style using HTTP is also defined in this standard. For that approach, HTTP GET operations SHALL be used to access the resources (GetResourceRepresentation) available on the server. The resource repesentations SHALL be equivalent to those that are obtained as a result of the GetCapabilites, GetTile and GetFeatureInfo operations in the procedure oriented architectural style. This approach will be described in clause 10.

8 WMTS using HTTP KVP encoding

WMTS servers may support service requests made using lists of parameters and their values defined as lists of Key-Value Pairs (KVP) and sent over HTTP using either GET or POST messages. Each pair is defined using the name of the parameter followed by an equals sign, '=' or ASCII character 61 (decimal), followed by the value given to the parameter, for example "service=WMTS". Parameter names for the operations are defined in subclause 7.1.2.1. For GET HTTP messages, the KVP lists are sent as part of the URL, as in the example of subclause 08.1.1. In POST HTTP messages, the KVP lists are sent in the message body, one pair per line.

Copyright © 2010 Open Geospatial Consortium Inc.

Any server wishing to support KVP requests SHALL declare its support by providing an OperationsMetadata section in its ServiceMetadata document with an Operation section for each supported operation and a section for each supported HTTP request type, GET or POST, in which KVP is declared as an AllowedValue, as explained in subclause 7.1.1.1.1. An example of this practice, declaring support for GetCapabilities operations using KVP with HTTP GET, can be seen in subclause 8.1.3.

8.1 GetCapabilities

WMTS servers may support KVP requests for a representation of the ServiceMetadata document by declaring support for and correctly handling GetCapabilities requests.

8.1.1 GetCapabilities request HTTP KVP encoding

A client performs a GetCapabilities operation using KVP over HTTP by sending a GET or POST HTTP message with the 'request' parameter set to 'GetCapabilities'.

8.1.2 GetCapabilities request HTTP KVP encoding example

To request a WMTS ServiceMetadata document, a client could issue the following KVP-encoded GetCapabilities operation request with minimal contents:

```
http://www.maps.bob/maps.cgi?service=WMTS&version=1.0.0&
request=GetCapabilities
```

8.1.3 GetCapabilities HTTP KVP encoding response

In response to a valid GetCapabilities request from a client, a WMTS server SHALL send a ServiceMetadata document which conforms with the XML schema defined in Annex B.

8.1.4 GetCapabilities HTTP KVP encoding response example

In response to a valid GetCapabilities operation request in KVP encoding, a WMTS server might generate a document that looks like the one in subclause 7.1.1.3.

The following fragment declares support for KVP encoded GetCapabilities operations using HTTP GET:

```
...
<ows:OperationsMetadata>
   <ows:Operation name="GetCapabilities">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://www.maps.bob/maps.cgi?">
               <ows:Constraint name="GetEncoding">
                  <ows:AllowedValues>
                     <ows:Value>KVP</ows:Value>
                  </ows:AllowedValues>
               </ows:Constraint>
            </ows:Get>
         </ows:HTTP>
      </ows:DCP>
```

 Copyright © 2010 Open Geospatial Consortium Inc.

```
</ows:Operation>
...
```

8.2 GetTile

WMTS servers may support KVP requests for representations of image Tiles by declaring support for and correctly handling GetTile requests.

8.2.1 GetTile request HTTP KVP encoding

Servers may implement HTTP GET transfer of the GetTile operation request, using KVP encoding. The KVP encoding of the GetTile operation request SHALL use the parameters specified in Table 29 which follows the abstract description specified in Table 22 above.

Table 29 — GetTile operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
`Service=WMTS`	Mandatory	Service type identifier
`Request=GetTile`	Mandatory	Operation name
`Version=1.0.0`	Mandatory	Standard and schema version for this operation
`Layer`	Mandatory	Layer identifier
`Style`	Mandatory	Style identifier
`Format`	Mandatory	Output format of tile
Sample dimensions [b]	Optional	Value allowed for this dimension
`TileMatrixSet`	Mandatory	TileMatrixSet identifier
`TileMatrix`	Mandatory	TileMatrix identifier
`TileRow`	Mandatory	Row index of tile matrix
`TileCol`	Mandatory	Column index of tile matrix

a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization SHALL be allowed in KVP encoding, see subclause 11.5.2 of OWS Common [OGC 06-121r3].

b Names for these parameters SHALL be the names indicated in the ServiceMetadata document. Typical examples are Time, Elevation and Band.

Parameters in a GetTile request may be specified in any order. However, in order to facilitate the caching mechanisms already available on the web, the parameters SHOULD be specified in the exact order that appears in Table 29.

8.2.2 GetTile request HTTP KVP encoding example

An example GetTile operation request KVP encoded for HTTP GET is:

```
http://www.maps.bob/maps.cgi?service=WMTS&request=GetTile&version=1.0.0&
layer=etopo2&style=default&format=image/png&TileMatrixSet=WholeWorld_CRS_84&
TileMatrix=10m&TileRow=1&TileCol=3
```

Copyright © 2010 Open Geospatial Consortium Inc.

8.2.3 GetTile HTTP KVP encoding response

The normal response to a valid GetTile operation request SHALL be a tile map that complies with the requested parameters and as described in subclause 7.2.1.

8.2.4 GetTile HTTP KVP encoding response example

A GetTile operation response for the GetTile request example in subclause 8.2.1 that corresponds with the ServiceMetadata document shown in subclause 7.1.1.3 is shown on Figure 12.

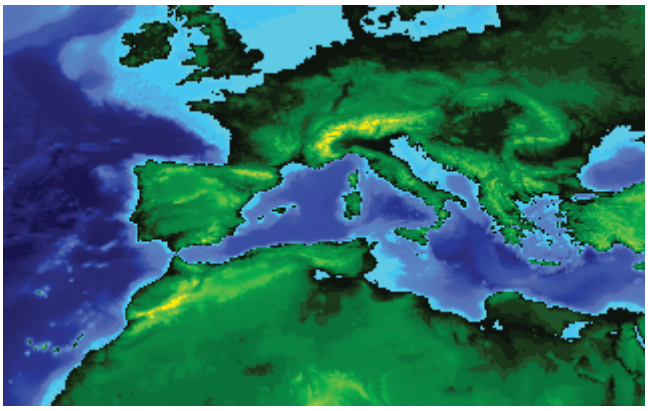


Figure 12 — GetTile response example

8.3 GetFeatureInfo

WMTS servers may support KVP requests for representations of documents furnishing information related to the features at particular pixel positions on particular image Tiles by declaring support for and correctly handling GetFeatureInfo requests.

8.3.1 GetFeatureInfo request HTTP KVP encoding

Servers may implement HTTP GET transfer of the GetFeatureInfo operation request, using KVP encoding. The KVP encoding of the GetFeatureInfo operation request SHALL follow the requirements for operation parameters specified in Table 30 that follows the abstract description specified in Table 25 above.

Table 30 — GetFeatureInfo operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
`Service=WMTS`	Mandatory	Service type identifier
`Request=GetFeatureInfo`	Mandatory	Operation name
`Version=1.0.0`	Mandatory	Standard and schema version for this operation

Copyright © 2010 Open Geospatial Consortium Inc.

Name and example [a]	Optionality and use	Definition and format
Sample dimensions [b]	Optional	Value allowed for this dimension
`Layer, Style, Format,` *Sample dimensions,* `TileMatrixSet, TileMatrix, TileRow, and TileCol`	Optionality and use of these parameters SHALL match those in the corresponding GetTile request described in Table 29	The values of these parameters SHALL match those in the corresponding GetTile request described in Table 29
`J`	Mandatory	Row index of a pixel in the tile
`I`	Mandatory	Column index of a pixel in the tile
`InfoFormat`	Mandatory	Output format of the retrieved information

a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization SHALL be allowed in KVP encoding, see subclause 11.5.2 of OWS Common [OGC 06-121r3].

b Names for these parameters SHALL be the names indicated in the ServiceMetadata document. Typical examples are Time, Elevation and Band.

Parameters in a GetFeatureInfo request may be specified in any order. However, in order to facilitate the caching mechanisms already available on the web, the parameters SHOULD be specified in the exact order that appears in Table 30

8.3.2 GetFeatureInfo request HTTP KVP encoding example

An example GetFeatureInfo operation request KVP encoded for HTTP GET is:

```
http://www.maps.bob/maps.cgi?service=WMTS&request=GetFeatureInfo&
version=1.0.0&layer=coastlines&style=default&format=image/png&
TileMatrixSet=WholeWorld_CRS_84&TileMatrix=10m&TileRow=1&TileCol=3&J=86&I=132&
InfoFormat=application/gml+xml; version=3.1
```

8.3.3 GetFeatureInfo HTTP KVP encoding response

The normal response to a valid GetFeatureInfo operation request SHALL be a FeatureInfo document as described in subclause 7.3.1.

8.3.4 GetFeatureInfo HTTP KVP encoding response example

A GetFeatureInfo operation response for the GetFeatureInfo request example in subclause 8.3.1 may look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<ReguralGriddedElevations xmlns="http://www.maps.bob/etopo2"
       xmlns:gml="http://www.opengis.net/gml"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.maps.bob/etopo2
              GetFeatureInfoExampleSchema.xsd">
   <featureMember>
       <GridPoint_etopo2>
          <elevation>503.0</elevation>
          <TileRow>1</TileRow>
          <TileCol>2</TileCol>
          <J>86</J>
          <I>132</I>
       </GridPoint_etopo2>
```

Copyright © 2010 Open Geospatial Consortium Inc.

```
    </featureMember>
</ReguralGriddedElevations>
```

8.4 Exceptions in HTTP KVP encoded operations

If an error is detected while processing an operation request encoded with KVP over HTTP, the WMTS server SHALL generate an ExceptionReport element (as defined in clause 8.5 of OWS Common [OGC 06-121r3]) and which conforms to the schemas described in Annex B.

9 WMTS using SOAP encoding

A WMTS server SHALL declare support for SOAP encoding for each operation by means of the OperationsMetadata section of its ServiceMetadata document as explained in subclause 7.1.1.1.1. An example of this practice for GetCapabilities operation can be seen in subclause 9.1.2. Annex F contains information on how to write a WSDL description document of the service.

9.1 GetCapabilities

9.1.1 GetCapabilities request SOAP encoding

Servers may also implement SOAP encoding using HTTP POST transfer of the GetCapabilities operation request, using SOAP version 1.2 encoding.

9.1.2 GetCapabilities request SOAP encoding example

To request a WMTS ServiceMetadata document, a client could issue the following SOAP encoded GetCapabilities operation request:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
    <soap:Body>
        <GetCapabilities service="WMTS"
                xmlns="http://www.opengis.net/ows/1.1">
            <AcceptVersions>
                <Version>1.0.0</Version>
            </AcceptVersions>
            <AcceptFormats>
                <OutputFormat>application/xml</OutputFormat>
            </AcceptFormats>
        </GetCapabilities>
    </soap:Body>
</soap:Envelope>
```

9.1.3 GetCapabilities SOAP encoding response

In response to GetCapabilities operation request in SOAP encoding, a WMTS server SHALL generate a document that looks like the one in subclause 7.1.1.3 wrapped in the SOAP version 1.2 envelope.

Copyright © 2010 Open Geospatial Consortium Inc.

9.1.4 GetCapabilities SOAP encoding response example

The following fragment remarks the SOAP envelope and the encoding of GetCapabilities response in the OperationsMetadata section:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <Capabilities version="1.0.0"
            xmlns="http://www.opengis.net/wmts/1.0"
            xmlns:ows="http://www.opengis.net/ows/1.1"
            xmlns:xlink="http://www.w3.org/1999/xlink"
            xmlns:gml="http://www.opengis.net/gml">
            ...
         <ows:OperationsMetadata>
            <ows:Operation name="GetCapabilities">
               <ows:DCP>
                  <ows:HTTP>
                     <ows:Post
               xlink:href="http://www.maps.bob/maps.cgi?">
                           <ows:Constraint name="PostEncoding">
                              <ows:AllowedValues>
                                 <ows:Value>SOAP</ows:Value>
                              </ows:AllowedValues>
                           </ows:Constraint>
                     </ows:Post>
                  </ows:HTTP>
               </ows:DCP>
            </ows:Operation>
            ...
         <WSDL xlink:role="http://schemas.xmlsoap.org/wsdl/1.0"
               xlink:show="none" xlink:type="simple"
               xlink:href="wmtsConcrete.wsdl"/>
      </Capabilities>
   </soap:Body>
</soap:Envelope>
```

9.2 GetTile

9.2.1 GetTile request SOAP encoding

Servers may also implement SOAP encoding using HTTP POST transfer of the GetTile operation request, using SOAP version 1.2 encoding.

9.2.2 GetTile request SOAP encoding example

An example of the SOAP encoding of the GetTile operation request equivalent of the request in subclause 8.2.1 is:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <GetTile service="WMTS" version="1.0.0"
               xmlns="http://www.opengis.net/wmts/1.0">
         <Layer>etopo2</Layer>
```

Copyright © 2010 Open Geospatial Consortium Inc.

```
            <Style>default</Style>
            <Format>image/png</Format>
            <TileMatrixSet> WholeWorld_CRS_84</TileMatrixSet>
            <TileMatrix>10m</TileMatrix>
            <TileRow>1</TileRow>
            <TileCol>3</TileCol>
        </GetTile>
    </soap:Body>
</soap:Envelope>
```

9.2.3 GetTile SOAP encoding response

The response of a successful SOAP-encoded GetTile operation request SHALL be an image with the MIME type specified by the Format parameter of the request, wrapped in the SOAP version 1.2 envelope. If the image is binary (such as is the case with image/png and image/jpeg images), it SHALL be base64 encoded and placed within the following XML element:

```
<element name="BinaryPayload">
    <complexType>
        <sequence>
            <element name="Format" type="ows:MimeType"/>
            <element name="PayloadContent" type="base64Binary"/>
        </sequence>
    </complexType>
</element>
```

The xs:base64Binary type and associated base64-encoding rules are defined in the XML Schema Part 2 W3C recommendation. MIME element includes the MIME type of the original BinaryPayload.

NOTE 1 The reason for using embedded encoded data instead of linking to an external source is to allow secured implementations.

Since the main part of the SOAP message is the base64 encoded binary content, base64 data SHOULD be enclosed inside a <![CDATA[ ]]>. This will prevent unnecessary parse of the base64 data resulting in a fast XML parse and validation.

NOTE 2 Current JavaScript XML parsers have limits on the length of the element content that are often too low to contain a base64 256x256 image in a single element. It as been seen that the use of <![CDATA[ ]]> is a workaround for this limitation.

9.2.4 GetTile SOAP encoding response example

An example of the SOAP response may look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
    <soap:Body>
        <wmts:BinaryPayload>
            <wmts:Format>image/png</wmts:Format>
            <wmts:BinaryContent>
                <!-- base64-encoded -->
        <![CDATA[R0lGODdh4AEOAfYAAOGW/9aM9MuC6L943LRu
        0KlkxJ1auJJQrIdGoXs8lXAyiWUofVkecU4U
```

 Copyright © 2010 Open Geospatial Consortium Inc.

```
            <!-- ... -->
        Ah0ianvIh+7Fb38oehcBI4NIiPdXhECyf4zY
        iGNnOq7FcfZTiUJ1hfyjVCW3bJ3IiYEAADs=]]>
            </wmts:BinaryContent>
        </wmts:BinaryPayload>
    </soap:Body>
</soap:Envelope>
```

9.3 GetFeatureInfo

9.3.1 GetFeatureInfo request SOAP encoding

Servers may also implement SOAP encoding using HTTP POST transfer of the GetFeatureInfo operation request, using SOAP version 1.2 encoding.

9.3.2 GetFeatureInfo request SOAP encoding example

An example of the SOAP encoding of the GetFeatureInfo operation request equivalent of the request in subclause 8.3.1 is:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
    <soap:Body>
        <GetFeatureInfo service="WMTS" version="1.0.0"
                  xmlns="http://www.opengis.net/wmts/1.0">
            <GetTile service="WMTS" version="1.0.0"
                  xmlns="http://www.opengis.net/wmts/1.0">
               <Layer>etopo2</Layer>
               <Style>default</Style>
               <Format>image/png</Format>
               <TileMatrixSet> WholeWorld_CRS_84</TileMatrixSet>
               <TileMatrix>10m</TileMatrix>
               <TileRow>1</TileRow>
               <TileCol>3</TileCol>
            </GetTile>
            <J>86</J>
            <I>132</I>
            <InfoFormat>application/gml+xml; version=3.1</InfoFormat>
        </GetFeatureInfo>
    </soap:Body>
</soap:Envelope>
```

9.3.3 GetFeatureInfo SOAP encoding response

The response of a successful SOAP-encoded GetFeatureInfo operation request SHALL be a document with the MIME type specified by the InfoFormat parameter of the request, wrapped in the SOAP version 1.2 envelope.

Since the GetFeatureInfo response format does not mandate any particular response format, the following flexible XML element that emphasizes the recommendation of GML Simple Features Profile level 0 response format is shown below:

```
<element name="FeatureInfoResponse">
    <complexType>
```

Copyright © 2010 Open Geospatial Consortium Inc.

```
        <choice>
            <element ref="gml:_FeatureCollection">
            <element ref="wmts:TextPayload">
            <element ref="wmts:BinaryPayload">
            <element name="AnyContent" type="anyType">
        </choice>
    </complexType>
</element>
```

9.3.4 GetFeatureInfo SOAP encoding response example

A GetFeatureInfo operation SOAP response for the GetFeatureInfo SOAP request example in subclause 9.3.2 may look like this in GML encoding:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="http://www.w3.org/2003/05/soap-envelope
    http://www.w3.org/2003/05/soap-envelope.xsd">
    <soap:Body>
        <FeatureInfoResponse>
            <ReguralGridedElevations
                xmlns="http://www.maps.bob/etopo2"
                xmlns:gml="http://www.opengis.net/gml"
                xsi:schemaLocation="http://www.maps.bob/etopo2
                wmtsGetFeatureInfo_response.xsd">
                <featureMember>
                    <GridPoint_etopo2>
                        <elevation>503.0</elevation>
                        <TileRow>1</TileRow>
                        <TileCol>2</TileCol>
                        <J>86</J>
                        <I>132</I>
                    </GridPoint_etopo2>
                </featureMember>
            </ReguralGridedElevations>
        </FeatureInfoResponse>
    </soap:Body>
</soap:Envelope>
```

A GetFeatureInfo operation SOAP response with the same information in HTML encoding may look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="http://www.w3.org/2003/05/soap-envelope
    http://www.w3.org/2003/05/soap-envelope.xsd">
    <soap:Body>
        <FeatureInfoResponse xmlns="http://www.opengis.net/wmts/1.0"
            xmlns:gml="http://www.opengis.net/gml"
            xsi:schemaLocation="http://www.opengis.net/wmts/1.0
            wmtsGetFeatureInfo_response.xsd">
```


Copyright © 2010 Open Geospatial Consortium Inc.

```
            <TextPayload>
                <Format>text/html</Format>
                <TextContent>
        <![CDATA[
        <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
        <HTML>
        <title>GetFeatureInfoResponse<title>
            <b>Elevation</b>503.0<br>
            <b>TileRow</b>1<br>
            <b>TileCol</b>2<br>
            <b>J</b>86<br>
            <b>I</b>132<br>
        </HTML>]]>
                </TextContent>
            </TextPayload>
        </FeatureInfoResponse>
    </soap:Body>
</soap:Envelope>
```

NOTE The use of `<![CDATA[ ]]>` is needed for embedded HTML data but will not be needed for XHTML.

A GetFeatureInfo operation SOAP response with the same information in arbitrary xml encoding may look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="http://www.w3.org/2003/05/soap-envelope
    http://www.w3.org/2003/05/soap-envelope.xsd">
    <soap:Body>
        <FeatureInfoResponse xmlns="http://www.opengis.net/wmts/1.0"
            xmlns:gml="http://www.opengis.net/gml"
            xsi:schemaLocation="http://www.opengis.net/wmts/1.0
            wmtsGetFeatureInfo_response.xsd">
            <AnyContent>
                <GridPoint_etopo2>
                    <elevation>503.0</elevation>
                    <TileRow>1</TileRow>
                    <TileCol>2</TileCol>
                    <J>86</J>
                    <I>132</I>
                </GridPoint_etopo2>
            </AnyContent>
        </FeatureInfoResponse>
    </soap:Body>
</soap:Envelope>
```

9.4 Exceptions in SOAP encoding

If an error is detected while processing an operation request encoded in a SOAP envelope, the WMTS server SHALL generate a SOAP 1.2 response message where the content of the Body element is a Fault element containing an ExceptionReport element (as defined in clause 8.5 of OWS Common [OGC 06-121r3]). This SHALL be done using the following XML fragment:

Copyright © 2010 Open Geospatial Consortium Inc.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
    <soap:Body>
        <soap:Fault>
            <soap:Code>
                <soap:Value>soap:Receiver</soap:Value>
            </soap:Code>
            <soap:Reason>
                <soap:Text>A server exception was encountered.</soap:Text>
            </soap:Reason>
            <soap:Detail>
                <ows:ExceptionReport
                    xmlns:ows="http://www.opengis.net/ows/1.1">
                    …
                </ows:ExceptionReport>
            </soap:Detail>
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

The Code element SHALL have the Value "soap:server" indicating that this is a server exception. The Reason element SHALL have the Text "A Server exception was encountered". This fixed string is used since the details of the exception SHALL be specified in the Detail element using an ows:ExceptionReport element.

10 WMTS using RESTful

A WMTS server that supports HTTP RESTful SHALL declare support for each resource by means of the ServiceMetadataURL (see Table 3) and the ResourceURL (see Table 6) elements of its ServiceMetadata document as explained in this clause. An example of this practice can be seen in the Annex D.

The first step in the resource oriented architectural style is to identify the resources and the relations between the resources. This version of WMTS standard identifies 3 resources: the service (ServiceMetadata), the map tiles (Tile) and the feature information related to a pixel of a tile (FeatureInfo). The RESTful approach provides a way to manipulate these resources via standard HTTP requests. This standard only defined the use of HTTP GET to download resource representations (that are equivalent to the ones that can be retrieved by the GetCapabilities, GetTile and GetFeatureInfo operations in the procedure oriented architectural style).

NOTE 1 Some RESTful literature calls this action of retrieval (downloading) of a resource using HTTP GET "GetResource" or "GetResourceRepresentation", the action to delete a resource using HTTP DELETE DeleteResource, etc. This is useful to clarify the action of the HTTP operation but note that there is no operation called explicitly GetResourceRepresentation operation in RESTful. The WMTS standard also adopts this notation allowing easy extension to other future RESTful actions.

The RESTful encoding for WMTS consists of a set of canonical URLs to the service metadata document, to tiles, and to FeatureInfo documents (one for each pixel). The service metadata document is the single entry point and all other URL endpoints can be obtained by analyzing the templates contained in the ResourceURL elements of each layer element in the ServiceMetadata document.

 Copyright © 2010 Open Geospatial Consortium Inc.

NOTE 2 Other ways of connecting resources were studied during WMTS standard elaboration and OWS-6 and URL templates seemed an interesting convenience that allows the description of thousands of tile URLs and millions of pixels FeatureInfo URLs with a few expressions rather than describing every required URL endpoint individually.

10.1 ServiceMetadata resource (mandatory in resource oriented architectural style)

10.1.1 GetResourceRepresentation request

In the resource oriented architectural style, clients will access the ServiceMetadata document simply by requesting a file to a standard HTTP server using the URL. The ServiceMetadata document SHALL have one or more `<ServiceMetadataURL>` elements indicating a URL where this document can be obtained.

The URL referencing a ServiceMetadata document can have any form but we recommend the following syntax and format:

```
{WMTSBaseURL}/1.0.0/WMTSCapabilities.xml
```

Clients can also specify the MIME type of the ServiceMetadata document by including an "Accept: " parameter in the HTTP header of the request.

10.1.2 GetResourceRepresentation request example

To request WMTS ServiceMetadata document in XML format, from a WMTS server that implements the resource oriented architectural style, a client could issue the following RESTful URL:

```
http://www.maps.bob/1.0.0/WMTSCapabilities.xml
```

10.1.3 ServiceMetadata representation

In response to a valid request for a ServiceMetadata representation from a client, a WMTS server SHALL send a ServiceMetadata document which conforms to the model described in 7.1.1. Servers SHALL be able to respond with a ServiceMetadata document in XML format (application/xml) that conforms to the XML schema described in Annex B but other formats are also allowed.

10.1.4 ServiceMetadata representation example

In response to a valid request for a ServiceMetadata representation from a client, a WMTS server might generate a document that looks like the one in subclause 7.1.1.3.

10.1.5 GetResourceRepresentation exception

If a client requests a document version or a format extension that is not available on a particular server, the server SHALL return an HTTP Error 404 (File Not Found).

Copyright © 2010 Open Geospatial Consortium Inc.

10.2 Tile resource (mandatory in resource oriented architectural style)

10.2.1 GetResourceRepresentation request

The ServiceMetadata document in the resource oriented architectural style MAY contain a list of Layer elements and each layer that is available to be retrieved in this architectural style SHALL have one or more <ResourceURL> elements with the "resourceType" attribute set to "tile" and a template attribute. In this RESTful approach the template attribute contains a URL template that can be converted to a URL by using a template processor and then get the expected tile in the format specified by the attribute "format" by requesting the resource with a standard HTTP GET.

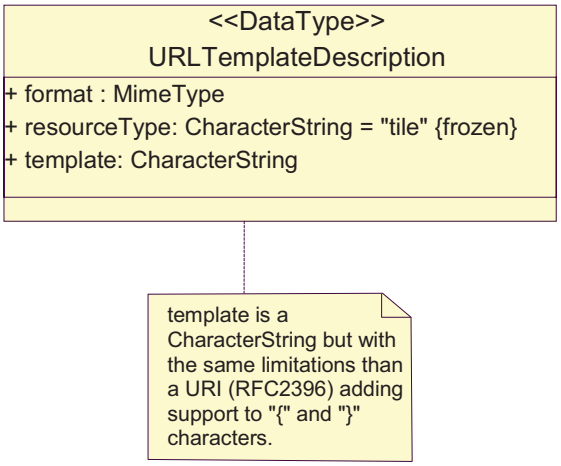


Figure 13 — URLTemplate for tile UML class diagram

Table 31 — Parts of the URLTemplate data structure for tiles

Names	Definition	Data type and values	Multiplicity and use
format `format`	Format of the resource representation that can be retrieved one resolved the URL template	ows:MimeType	One (mandatory)
resource Type `resource Type`	Resource type to be retrieved	Character String type, not empty SHALL contain "tile"	One (mandatory)
template `template`	URL template [a].	Character String type but with the same limitations than a URI (RFC2396) adding support to "{" and "}" characters.	One (mandatory)
a A template processor will apply the rules in Table 32 to get a URL to a resource.			

A template processor is a program or library that runs on the client side and converts a URL template into a URL. It will have to process the URL template that contains variable names marked off in matching braces ('{', '}') and substitute them with the corresponding

 Copyright © 2010 Open Geospatial Consortium Inc.

valid value. The template processor SHALL support the following variable names: "Style", "TileMatrixSet", "TileMatrix", "TileRow", "TileCol" and any dimension identifier that has been defined for this layer in the element ./Dimension/ows:Identifier. The template processor SHALL substitute the variable names by the values of the elements as shown in Table 32.

Possible values and ranges of the variables in the URL template can be extracted from the SeviceMetadata document parameters. The following table lists all the possible variable names, their description, possible values and multiplicity. Assuming a ServiceMetadata in XML format the possible values are also given using XPath expressions that point to the ServiceMetadata document. When a relative XPath is used, it is relative to its layer element of the ServiceMetadata document.

When a variable has only one possible value for this layer, the use of the direct value instead of the variable is recommended on the URL template.

Table 32 — URL template variables and possible values for tile

URL template variable	Meaning	Possible values	Multiplicity
`"style"`	Style identifier	identifier in Table 7 `./style/ows:Identifier`	One (mandatory) [a]
`./Dimension/ ows:Identifie r`	Dimension value	identifier in Table 9 `./Dimension[ows:Identifier={./Dimension/ ows:Identifier}]/Value`	One for each dimension available (mandatory if there are dimensions defined)
`"TileMatrix Set"`	tile matrix set identifier	identifier in Table 6 `./ TileMatrixSetLink/TileMatrixSet`	One (mandatory) [a]
`"TileMatrix"`	tile matrix identifier	identifier in Table 14 `/Capabilities/Contents/TileMatrixSet[ ows:Identifier={TileMatrixSet}]/TileMatrix / ows:Identifier`	One (mandatory) [a]
`"TileRow"`	row index of tile matrix	If TileMatrixSetLimits is present, see Table 10, (`./TileMatrixSetLimits)`, SHALL be any integer value between MinTileRow and MaxTileRow in Table 12 (both included) (`./tileMatrixSetLimits/tileMatrixLimits[./ TileMatrix={TileMatrix}]/MinTileRow` and `./tileMatrixSetLimits/tileMatrixLimits[./TileMatrix={TileMatrix}]/MaxTileRow`). else SHALL be any integer value between 0 and MatrixHeight – 1, see Table 14, (both included) (0 and `/Capabilities/Contents/TileMatrixSet[ows:Identifier={TileMatrixSet}]/TileMatrix[ows:Identifier={TileMatrix}]/MatrixHeight - 1`)	One (mandatory) [a]

Copyright © 2010 Open Geospatial Consortium Inc.

URL template variable	Meaning	Possible values	Multiplicity
"TileCol"	col index of tile matrix	If TileMatrixSetLimits is present, see Table 10, (./TileMatrixSetLimits), SHALL be any integer value between MinTileCol and MaxTileCol in Table 12 (both included) (./tileMatrixSetLimits/tileMatrixLimits[./TileMatrix={TileMatrix}]/MinTileCol and ./tileMatrixSetLimits/tileMatrixLimits[./TileMatrix={TileMatrix}]/MaxTileCol) else SHALL be any integer value between 0 and MatrixWidth – 1, see Table 14, (both included) (0 and /Capabilities/Contents/TileMatrixSet[ows:Identifier={TileMatrixSet}]/TileMatrix[ows:Identifier={TileMatrix}]/ MatrixWidth – 1)	One (mandatory) [a]
a When a variable has only one possible value for this layer the use of the direct value instead of the variable is recommended.			

Any possible order of the variables and values in the URL template is valid. Neverdeless we recommend the following order:

style, firstDimension, ..., lastDimension, TileMatrixSet, TileMatrix, TileRow and TileCol

NOTE 1 It is not necessary for a file system with this particular structure to actually exist on the server. A WMTS server implementation is free to parse the request URL itself and either remap the request to a different internal directory structure or generate the response on the fly (at the expense of speed).

NOTE 2 This syntax and processing is in-line with the pre-existing templating schemas present in OpenSearch, WSDL and WADL and is a simplification of the description made by the IETF Network Working Group in the Internet-Draft called "URI Template draft-gregorio-uritemplate-03".

10.2.2 GetResourceRepresentation request example

An example of a tile URL for a resource representation in RESTful HTTP GET equivalent of the request in subclause 8.2.1 is:

```
http://www.maps.bob/etopo2/default/WholeWorld_CRS_84/10m/1/3.png
```

It corresponds to the following ResourceURL element:

```
<ResourceURL format="image/png" resourceType="tile"
template="http://www.maps.bob/etopo2/default/
{TileMatrixSet}/{TileMatrix}/{TileRow}/{TileCol}.png">
```

10.2.3 Tile representation

In response to a valid request for a Tile representation from a client, a WMTS server SHALL send either an image representation of the tile or a reference to an image, as stated in subclause 7.2.1. An image is the most typical representation but representations in other formats are also allowed.

10.2.4 Tile representation example

The response could be the same image shown in subclause 8.2.4.

 Copyright © 2010 Open Geospatial Consortium Inc.

10.2.5 GetResourceRepresentation exception

If the response of a GetResourceRepresentation request for a tile is unsuccessful, the server SHALL return an HTTP error code that SHOULD be accompanied by an XML ExceptionReport document as defined in subclause 8.5 of OWS Common [OGC 06-121r3] and subclause 7.2.2.2 of this document. Determination of what error conditions map to which HTTP error codes is up to the discretion of the server. If the error condition is due to a malformed request or the resource does not exist (for instance in a request URL with illegal path values), the HTTP error code returned SHOULD be HTTP Error 404 (File Not Found).

NOTE For a server where the tile resource files actually exists and there is no specific application capable of generating specific responses on the fly, HTTP error code SHALL be expected but an XML ExceptionReport document cannot be expected.

10.3 FeatureInfo resource (optional in resource oriented architectural style)

10.3.1 GetResourceRepresentation request

The ServiceMetadata document in the resource oriented architectural style may contain a list of Layer elements and each layer that is available to be retrieved in this architectural style and is queryable SHALL have one or more <ResourceURL> elements with the "resourceType" attribute set to "FeatureInfo" and a template attribute. In this RESTful approach the template attribute contains a URL template that can be converted to a URL by using a template processor and then get the expected FeatureInfo in the format specified by the attribute "format" by requesting the resource with a standard HTTP GET.

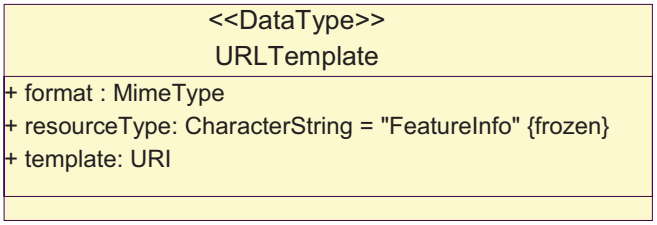


Figure 14 — URLTemplate for the FeatureInfo UML class diagram

Table 33 — Parts of the URLTemplate data structure for FeatureInfo

Names	Definition	Data type and values	Multiplicity and use
format `format`	Format of the resource representation that can be retrieved or resolved from the URL template	ows:MimeType	One (mandatory)
resource Type `resource Type`	Resource type to be retrieved	Character String type, not empty SHALL contain "FeatureInfo"	One (mandatory)
template `template`	URL template [a].	URI	One (mandatory)
a A template processor will apply the rules in Table 34 to get a URL to a resource.			

Copyright © 2010 Open Geospatial Consortium Inc.

A template processor is a program or library that runs on the client side and converts a URL template into a URL. It will have to process the URL template that contains variable names marked off in matching braces ('{', '}') and substitute them with the corresponding valid value. The template processor SHALL support the following variable names: "Style", "TileMatrixSet", "TileMatrix", "TileRow", "TileCol", "J", "I", and any dimension identifier that has been defined for this layer in the element ./Dimension/ows:Identifier. The template processor SHALL substitute the variable names by the values shown in Table 34.

Possible values and ranges of the variables in the URL template can be extracted from the SeviceMetadata document parameters. The following table lists all the possible variable names, their description, possible values and multiplicity. Assuming a ServiceMetadata in XML format the possible values are also given using XPath expressions that point to the ServiceMetadata document. When a relative XPath is used, it is relative to its layer element of the ServiceMetadata document.

When a variable has only one possible value for this layer, the use of the direct value instead of the variable is recommended on the URL template.

Table 34 — URL template variables and possible values for FeatureInfo

URL template variable	Meaning	Possible values	Multiplicity
`"style"`	Style identifier	identifier in Table 7 `./style/ows:Identifier`	One (mandatory) [a]
`./Dimension/ ows:Identifie r`	Dimension value	identifier in Table 9 `./Dimension[ows:Identifier={./Dimension/ ows:Identifier}]/Value`	One for each dimension available (mandatory if there are dimensions defined)
`"TileMatrix Set"`	tile matrix set identifier	identifier in Table 6 `./ TileMatrixSetLink/TileMatrixSet`	One (mandatory) [a]
`"TileMatrix"`	tile matrix identifier	identifier in Table 14 `/Capabilities/Contents/TileMatrixSet[ ows:Identifier={TileMatrixSet}]/TileMatrix / ows:Identifier`	One (mandatory) [a]

 Copyright © 2010 Open Geospatial Consortium Inc.

URL template variable	Meaning	Possible values	Multiplicity
"TileRow"	row index of tile matrix	If TileMatrixSetLimits is present, see Table 10, (./TileMatrixSetLimits), SHALL be any integer value between MinTileRow and MaxTileRow in Table 12 (both included) (./tileMatrixSetLimits/tileMatrixLimits[./TileMatrix={TileMatrix}]/MinTileRow and ./tileMatrixSetLimits/tileMatrixLimits[./TileMatrix={TileMatrix}]/MaxTileRow). else SHALL be any integer value between 0 and MatrixHeight – 1, see Table 14, (both included) (0 and /Capabilities/Contents/TileMatrixSet[ows:Identifier={TileMatrixSet}]/TileMatrix[ows:Identifier={TileMatrix}]/MatrixHeight – 1)	One (mandatory) [a]
"TileCol"	col index of tile matrix	If TileMatrixSetLimits is present, see Table 10, (./TileMatrixSetLimits), SHALL be any integer value between MinTileCol and MaxTileCol in Table 12 (both included) (./tileMatrixSetLimits/tileMatrixLimits[./TileMatrix={TileMatrix}]/MinTileCol and ./tileMatrixSetLimits/tileMatrixLimits[./TileMatrix={TileMatrix}]/MaxTileCol) else SHALL be any integer value between 0 and MatrixWidth – 1, see Table 14, (both included) (0 and /Capabilities/Contents/TileMatrixSet[ows:Identifier={TileMatrixSet}]/TileMatrix[ows:Identifier={TileMatrix}]/ MatrixWidth – 1)	One (mandatory) [a]
"J"	Pixel row index in a tile	SHALL be any integer value between 0 and TileHeight-1 (see Table 14) (both included) (0 and /Capabilities/Contents/TileMatrixSet[ows:Identifier={TileMatrixSet}]/TileMatrix[ows:Identifier={TileMatrix}]/TileHeight – 1)	One (mandatory) [a]
"I"	Pixel column index in a tile	SHALL be any integer value between 0 and TileWidth -1 (see Table14) (both included) (0 and /Capabilities/Contents/TileMatrixSet[ows:Identifier={TileMatrixSet}]/TileMatrix[ows:Identifier={TileMatrix}]/TileWidth – 1)	One (mandatory) [a]
a When a variable has only one possible value for this layer the use of the direct value instead of the variable is recommended.			

Any possible order of the variables and values in the URL template is valid. Neverdeless we recommend the following order:

style, firstDimension, ..., lastDimension, TileMatrixSet, TileMatrix, TileRow, TileCol, J and I.

NOTE It is highly improbable that all of the GetFeatureInfo response file URLs physically exist in a practical implementation, so for a GetFeatureInfo-enabled server it seems inevitable that a RESTful server implementation would generate them on the fly or cache them.

Copyright © 2010 Open Geospatial Consortium Inc.

10.3.2 GetResourceRepresentation request example

An example FeatureInfo request for a resource representation in RESTful HTTP GET equivalent of request in subclause 8.2.1 is

```
http://www.maps.bob/etopo2/
default/WholeWorld_CRS_84/10m/1/3/86/132.xml
```

It corresponds to the following ResourceURL element:

```
<ResourceURL format="application/gml+xml; version=3.1"
resourceType="FeatureInfo"
template="http://www.maps.bob/etopo2/default/
{TileMatrixSet}/{TileMatrix}/{TileRow}/{TileCol}/{J}/{I}.xml">
```

10.3.3 FeatureInfo representation

In response to a valid request for a FeatureInfo representation from a client, a WMTS server SHALL send a document with information related to a particular pixel of a tile or a reference to resources that represents the geographic data portrayed on that area, as stated in subclause 7.3.1. An XML document conforming GML Simple Features Profile [06-049r1] is the most typical representation but other representations and formats are also allowed.

10.3.4 FeatureInfo representation as an XML document example

The response could be the same xml document shown in subclause 8.3.4.

10.3.5 GetResourceRepresentation exception

If the response of a GetResourceRepresentation request for a FeatureInfo is unsuccessful the server SHALL return an HTTP error code that SHOULD be accompanied by an XML ExceptionReport document as defined in subclause 8.5 of OWS Common [OGC 06-121r3] and subclause 7.3.2.2 of this document. Determination of what error conditions map to which HTTP error codes is up to the discretion of the server. If the error condition is due to a malformed request or the resource does not exist (for instance in a request URL with illegal path values), the HTTP error code returned SHOULD be HTTP Error 404 (File Not Found).

11 Recommendations to improve interoperability and performance.

For maximum interoperability, this WMTS standard makes the following recommendations.

11.1 Server and Client support for KVP, SOAP and RESTful

This specification defines 3 interfaces KVP, SOAP and RESTful. Clients and servers are encouraged to support as many interfaces as possible to improve interoperability. The minimum recommended support is:

 Copyright © 2010 Open Geospatial Consortium Inc.

A WMTS client SHOULD support both KVP and RESTful. SOAP support is optional.

A WMTS server SHOULD support KVP and/or RESTful. SOAP support is optional.

11.2 A standard set of scales

It is strongly recommended that WMTS servers offer their Layers using, where possible, the well-known scale sets defined in the Annex E and reference them in the service metadata document in the TileMatrixSetDef section. This would aid interoperability between systems and allow users to "mash-up" suitable services.

11.3 A standard image format and FeatureInfo document response

It is recommended that servers offer Tiles in the image/png and image/jpeg file formats. The image/png image format is good for categorical maps and image/jpeg is better for imagery but since image/jpeg does not support transparency, image/png may be used for images. It is recommended that WMTS Clients support both. Since a GetTile operation can serve only one tile at a time, it is important that clients have the ability to support transparency and also be able to overlap tiles from the same geographical area.

It is recommended that FeatureInfo documents be offered in the MIME type format " application/gml+xml; version=3.1" and to follow GML Simple Features Profile [06-049r1]. See subclause 7.3.1.

11.4 Number of TileMatrixSets and TileMatrixSetLimits

In a server where all layers cover almost the same area, or layers can cover the same area in the future it is recommended to use the same TileMatrixSet for all layers and to use TileMatrixSetLimits to inform clients about the tile matrix indices currently available.

This will avoid changing all tile indices if some future change extends the area covered.

11.5 Cacheble resources

One of the main goals of this standard is to provide a better cache support for repetitive requests, helping WMTS servers to save resources and perform better. Web caching occurs at several levels, but for caching to occur at any of these levels caching mechanisms need to know that resources are cachable. Servers should include appropriate caching information (expiration date) in the WMTS server responses to avoid impacting performance. In a RESTful file based server where tiles are pre-rendered and stored in directories that are directly accessed by generic internet servers, the expiration date of the data can be configured as a property of this directory.

Internally this is achieved by means of the proper HTTP control headers:

HTTP 1.0, uses the "Expires" header. This header indicates an expiration date. If your data is guarantied to be static, or you know when the data is going to be updated, you can use a convenient future date in the Expires header.

Copyright © 2010 Open Geospatial Consortium Inc.

HTTP 1.1, uses the "Cache-control" header. This header indicates a period of time to cache the data before expiration. If your data is guarantied to be static, or you know when the data is going to be updated, you can use a convenient period of time in the Cache-control header.

 Copyright © 2010 Open Geospatial Consortium Inc.

Annex A
(normative)

Abstract test suite

A.1 Introduction

This abstract test suite specifies at a high level how server and client implementations of this standard SHALL be tested for conformance to this standard. The framework for such abstract test suites is specified in ISO 19105: Geographic information – Conformance and testing, especially Clauses 7 and 9.

An abstract test suite contains multiple abstract tests, grouped into one or more test modules. This abstract test suite consists of two top-level test modules:

a) Client test module – Abstract tests for checking conformance of client implementations with the requirements of this standard that are normatively referenced by this Implementation Specification.

b) Server test module – Abstract tests for checking conformance of server implementations with the requirements of this standard that are normatively referenced by this Implementation Specification.

Any of these modules could contain lower-level test modules. At this time, only the Server test module contains lower-level test modules, namely:

a) All operations implemented test module – Abstract tests for checking server properties that are common to all operations implemented.

b) GetCapabilities, GetTile and GetFeatureInfo operation test module – Abstract tests for checking server properties that are specific to an operation.

In the client and server test modules, all operations specified and implemented SHALL be tested, including KVP HTTP GET, and SOAP HTTP POST transfer and RESTFul HTTP GET transfer of each operation request. In the standard test module, all operations specified SHALL be checked, including KVP HTTP GET, SOAP HTTP POST and RESTFul HTTP GET transfer of operation requests. And all operation request and response parameters specified or implemented SHALL be tested. Of course, some operations, transfer methods, and parameters are specified as optional implementation by servers. Any optional item not implemented by a server SHALL not be tested. Also, items not implemented by a client SHALL not be tested.

Copyright © 2010 Open Geospatial Consortium Inc.

A.2 Client test module

A.2.1 GetCapabilities operation request

a) Test Purpose: Verify that a client satisfies all requirements for a GetCapabilities operation request.

b) Test Method: Generate an adequate sample of GetCapabilities operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 7.1.2.1

d) Test Type: Basic

A.2.2 GetTile operation request

a) Test Purpose: Verify that a client satisfies all requirements for a GetTile operation request.

b) Test Method: Generate an adequate sample of GetTile operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 7.2.2

d) Test Type: Basic

A.2.3 Contiguous GetTile operation requests

a) Test Purpose: Verify that a client is capable of generating contiguous GetTile operation requests.

b) Test Method: Generate adequate samples of GetTile operation requests from the client, for tiles that are contiguous, and verify that the client is able to show them without any discontinuity.

c) Reference: Subclause 7.2.1 and 7.2.2

d) Test Type: Capability

A.2.4 Overlay and transparency in GetTile operation request

a) Test Purpose: Verify that a client is capable to generating overlaying GetTile operation requests, each one for a different layer.

b) Test Method: Generate adequate samples of GetTile operation requests from the client, for tiles that overlays and verify that the client is able to show them correctly overlaid and with a transparency when NODATA is present on the upper images when the image format allows that.

c) Reference: Subclauses 7.2.1,7.2.2 and 11.3

 Copyright © 2010 Open Geospatial Consortium Inc.

d) Test Type: Capability

A.2.5 Different TileMatrixSets in GetTile operation requests

a) Test Purpose: Verify that a client can correctly overlay layers with layers having different TileMatrixSets.

b) Test Method: Generate adequate samples of GetTile operation requests from the client of tiles from layers with different TileMatrixSets, and verify that client is able to show them correctly overlaid.

c) Reference: Subclauses 7.2.1 and 7.2.2

d) Test Type: Capability

A.2.6 Optional GetFeatureInfo operation request

a) Test Purpose: Verify that a client satisfies all requirements for a GetFeatureInfo operation requests.

b) Test Method: Generate an adequate sample of GetFeatureInfo operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 7.3.2.1

d) Test Type: Basic

A.2.7 GetTile and GetFeatureInfo operation request form ServiceMetadata content.

e) Test Purpose: Verify that a client is able to parse a GetCapabilities response and generate GetTile and GetFeatureInfo operation request.

f) Test Method: Generate an adequate sample of GetCapabilities operation requests from the client, and verify that the client is able to generate valid GetTile and GetFeatureInfo requests conforming with the content section of the ServiceMetadata document.

g) Reference: Subclause 7.1.1

h) Test Type: Basic

A.3 Server test module

A.3.1 All operations implemented test module

A.3.1.1 HTTP protocol usage

a) Test purpose: Verify that the rules and conventions governing the use of HTTP are observed.

Copyright © 2010 Open Geospatial Consortium Inc.

b) Test method: TBD

c) Reference: RFC 2616 (Hypertext Transfer Protocol -- HTTP/1.1). See <http://www.ietf.org/rfc/rfc2616>.

d) Test type: Capability

A.3.1.2 Accept HTTP GET and POST transferred operation requests

a) Test Purpose: Verify that a server accepts HTTP GET and/or HTTP POST transferred requests for each operation.

b) Test Method: Submit HTTP GET and/or HTTP POST transferred requests for each operation. Verify that the server accepts and responds to these requests as specified and implemented. Check that the server accepts at least one HTTP GET or HTTP POST transfer of requests for each operation.

c) Reference: Clause 8, and 9

d) Test Type: Capability

A.3.1.3 Handle KVP-encoded operation requests

a) Test Purpose: Verify that a server handles all parameter names in a KVP-encoded operation request in a capitalization- and sequence-insensitive manner.

b) Test Method: Submit KVP-encoded GetCapabilities and other operation requests containing parameter names using various combinations of cases, with a variety of parameter sequences. Verify that the server provides the same response when the same parameter names use different cases and combinations of cases.

c) Reference: Clause 8

d) Test Type: Capability

A.3.1.4 Handle SOAP-encoded operation requests

a) Test Purpose: Verify that a server handles all parameters in a XML-encoded operation request in a name-capitalization and parameter-sequence sensitive manner.

b) Test Method: Submit SOAP-encoded GetCapabilities request and other operation requests containing parameters using correct and incorrect name capitalizations and parameter sequences. Verify that the server accepts all correct requests, and returns ExceptionReport messages for all incorrect requests.

c) Reference: Clause 9

d) Test Type: Capability

A.3.1.5 Handle HTTP GET RESTful -encoded operation requests

a) Test Purpose: Verify that a server handles a URL service metadata request.

 Copyright © 2010 Open Geospatial Consortium Inc.

b) Test Method: Request a Service metadata URL and other URL resources using correct and incorrect URLs. Verify that the server respond with the right resource to correct URLs, and a returns HTTP errors for invalid URLs.

c) Reference: Clause 10

d) Test Type: Capability

A.3.1.6 KVP and SOAP HTTP response status code

a) Test purpose: Verify that a service request which generates an exception produces a response that contains 1) a correct service exception report, and 2) the correct status code indicating the error.

b) Test method: Check the response code in the Status-Line and the message body. Pass if the response code is either 4xx (Client error) or 5xx (Server error) and the body contains a service exception report. Fail otherwise.

c) Reference: RFC 2616, clause 11 and Subclauses 7.1.2.2, 7.2.2.2 and 7.3.2.2

d) Test type: Capability

A.3.2 GetCapabilities operation request test module (Procedure Oriented Architectural Style)

A.3.2.1 Accept HTTP GET transferred operation requests

a) Test Purpose: Verify that a server accepts at least HTTP GET transferred requests for the GetCapabilities operation.

b) Test Method: Submit HTTP GET transferred requests for the GetCapabilities operation. Verify that the server accepts and responds to these requests as specified.

c) Reference: Subclause 8.1.1

d) Test Type: Capability

A.3.2.2 GetCapabilities operation response

a) Test Purpose: Verify that a server satisfies all requirements of the GetCapabilities operation response.

b) Test Method: Make several GetCapabilities requests including a variety of input parameters. Verify that the specified correct response is returned to each request.

c) Reference: Subclause 8.1.3

d) Test Type: Capability

A.3.2.3 Version negotiation

a) Test Purpose: Verify that a server satisfies the requirements for version negotiation.

Copyright © 2010 Open Geospatial Consortium Inc.

b) Test Method: Submit GetCapabilities operation requests containing version numbers lower than, higher than, and equal to the version supported by the server. Verify that the server responses are in accord with the specified rules for version negotiation.

c) Reference: Subclause 7.3.2 of OWS Common [OGC 06-121r3]

d) Test Type: Capability

A.3.2.4 Format selection

a) Test Purpose: Verify that a server satisfies the requirements for format selection, if the server implements the AcceptFormats request parameter.

b) Test Method: Submit GetCapabilities operation requests containing supported and unsupported values for the AcceptFormats parameter. Verify that the server responses are in accord with the specified rules for format selection.

c) Reference: Subclause 7.3.5 of OWS Common [OGC 06-121r3]

d) Test Type: Capability

A.3.2.5 Handling updateSequence parameter

a) Test Purpose: Verify that a server satisfies the requirements for generating and using the updateSequence parameter, if the server implements the AcceptFormats request parameter.

b) Test Method: Submit GetCapabilities operation requests containing correct and incorrect values of the AcceptFormats parameter. Verify that the server provides the specified correct response to each request.

c) Reference: Subclause 7.3.4 of OWS Common [OGC 06-121r3]

d) Test Type: Capability

A.3.2.6 Section selection

a) Test Purpose: Verify that a server satisfies the requirements for using the Sections parameter, if the server implements the Sections request parameter.

b) Test Method: Submit GetCapabilities operation requests containing various values and combinations of values of the Sections parameter. Verify that the server provides the specified correct response to each request

c) Reference: Subclause 7.1.2.1 and Table 18

d) Test Type: Capability

A.3.3 ServiceMetadata resource test module (Resource Oriented Architectural Style)

A.3.3.1 Accept HTTP GET transferred operation requests

a) Test Purpose: Verify that a server sends a correct ServiceMetadata resource.

 Copyright © 2010 Open Geospatial Consortium Inc.

b) Test Method: Perform HTTP GET requests to the ServiceMetadata URL Verify that the specified correct response is returned to each request.

c) Reference: Subclause 10.1.1

d) Test Type: Capability

A.3.4 ServiceMetadata response

A.3.4.1 XML well formated

a) Test purpose: Verify that the ServiceMetadata document is a valid xml document.

b) Test method: Submit GetCapabilities operation requests and verify that the returned document is a well formed xml document.

c) Reference: Subclause 7.1.1

d) Test type: Capability

A.3.4.2 XML references the normative schema

a) Test purpose: Verify that the normative content of the schema document referred to by the schemaLocation attribute in the ServiceMetadata document is identical to the normative content of the on-line schema referred to in the Annex B.

b) Test method: Pass if the normative content of the schema document referred to by the schemaLocation attribute in the ServiceMetadata document is identical to the normative content of the on-line schema referred to in the Annex B.1.

c) Reference: Subclause 7.1.1

d) Test type: Capability

A.3.4.3 XML validates against the schema

a) Test purpose: Verify that the response to a GetCapabilities request validates against the schema(s) provided with the schemaLocation attribute.

b) Test method: Pass if the response to a GetCapabilities request validates against the schema(s) provided with the schemaLocation attribute.

c) Reference: Subclause 7.1.1

d) Test type: Capability

A.3.4.4 OnLineResource is an only resource prefix

a) Test purpose: Verify that each OnlineResource URL intended for HTTP Get requests in the ServiceMetadata document is a URL prefix.

Copyright © 2010 Open Geospatial Consortium Inc.

b) Test method: Pass if each OnlineResource URL intended for HTTP Get requests in the ServiceMetadata document is a URL prefix.

c) Reference: Subclause 7.1.1

d) Test type: Capability

A.3.4.5 XML format for GetCapabilities

a) Test purpose: Verify that the server advertises the application/xml format for the GetCapabilities operation.

b) Test method: Pass if the server advertises the application/xml format for the GetCapabilities operation.

c) Reference: Subclause 7.1.1

d) Test type: Capability

A.3.4.6 ows:Constraint GetEncoding

a) Test Purpose: Verify that a server satisfies the requirements for using the ows:Constraint GetEncoding parameter, if the server implements the Sections request parameter.

b) Test Method: Verify that the server provides a Service Metadata document that includes ows:Constraint GetEncoding information on OperationsMetadata section. Verify that the server is able to respond the encodings specified.

c) Reference: Subclause 7.1.1

d) Test Type: Capability

A.3.4.7 ows:Constraint PostEncoding

a) Test Purpose: Verify that a server satisfies the requirements for using the ows:Constraint PostEncoding parameter, if the server implements the Sections request parameter.

b) Test Method: Verify that the server provides a Service Metadata document that includes ows:Constraint PostEncoding information on OperationsMetadata section. Verify that the server is able to respond the encodings specified.

c) Reference: Subclause 7.1.1

d) Test Type: Capability

A.3.4.8 Layer identifiers

a) Test purpose: Verify that the layer identifiers are different.

b) Test method: Pass if all the layers have different non empty identifiers.

 Copyright © 2010 Open Geospatial Consortium Inc.

c) Reference: Subclause 7.1.1.1.2 and Table 6

d) Test type: Capability

A.3.4.9 Layer LegendURL are correct resources

a) Test purpose: Verify that the metadata for each of the LegendURL resources is correct.

b) Test method: Pass if all the submodules and subtests pass.

c) Reference: Subclause 7.1.1.1.2 and Table 8

d) Test type: Capability

A.3.4.10 Layer LegendURL correct Format

a) Test purpose: Verify that the MIME-type returned for the LegendURL resource is the advertised format.

b) Test method: Pass if the MIME-type returned for the LegendURL resource is the advertised format.

c) Reference: Subclause 7.1.1.1.2 and Table 11

d) Test type: Capability

A.3.4.11 Layer LegendURL correct sizes

a) Test purpose: Verify that the size of the LegendURL resource is the advertised width and the advertised height.

b) Test method: Pass if the size of the LegendURL resource is the advertised width and the advertised height.

c) Reference: Subclause 7.1.1.1.2 and Table 8

d) Test type: Capability

A.3.4.12 Layer TileMatrixSet is valid

a) Test purpose: Verify that Layer TileMatrixSet contains a correct identifier.

b) Test method: Pass if Layer TileMatrixSet value is equal to a TileMatrixSet identifier in the content section.

c) Reference: Subclause 7.1.1.1.2 and Table 8

d) Test type: Capability

A.3.4.13 TileMatrixSet Identifier

a) Test purpose: Verify that TileMatrixSet identifiers are correct.

b) Test method: Pass if all TileMatrixSet have different non empty identifiers.

Copyright © 2010 Open Geospatial Consortium Inc.

c) Reference: Subclause 7.1.1.1.2 and Table 13

d) Test type: Capability

A.3.4.14 TileMatrix Identifier

a) Test purpose: Verify that TileMatrix identifiers are correct.

b) Test method: Pass if all TileMatrix in a TileMatrixSet have different non empty identifiers.

c) Reference: Subclause 7.1.1.1.2 and Table 14

d) Test type: Capability

A.3.4.15 TileMatrixSet ScaleDenominators

a) Test purpose: Verify that ScaleDenominator values are correct.

b) Test method: Pass if ScaleDenominators in a TileMatrixSet have different non empty values.

c) Reference: Subclause 7.1.1.1.2 and Table 14

d) Test type: Capability

A.3.4.16 TileMatrixSet WellKnownScaleSet

a) Test purpose: Verify that a WellKnownScaleSet is compatible with ScaleDenominator values.

b) Test method: When a WellKnownScaleSet is advertised, there has to be a TileMatrix for each different ScaleDenominator with values starting from the largest scale denominator in the WellKnownScaleSet table and all intermediate scales denominators down to some ScaleDenominator minimum value for this Layer.

c) Reference: Subclause 7.1.1.1.2 and Table 14

d) Test type: Capability

A.3.4.17 Theme LayerRef is valid

a) Test purpose: Verify that Theme LayerRef contains a correct identifier.

b) Test method: Pass if each Theme LayerRef value is equal to a Layer identifier in the content section.

c) Reference: Subclause 7.1.1.1.3 and Table 15

d) Test type: Capability

 Copyright © 2010 Open Geospatial Consortium Inc.

A.3.5 Tile request test module

A.3.5.1 GetTile Layer

a) Test purpose: Verify that when a request contains a Layer incorrect value, then the server throws an exception.

b) Test method: When a request contains a Layer that is not advertised in the ServiceMetadata document, then the server throws an exception (code= InvalidParameterValue) in Procedure Oriented Architectural Style.

c) Reference: Subclause 7.2.2.2 and Table 26

d) Test type: Capability

A.3.5.2 Tile ResourceURL template

a) Test Purpose: Verify that a client supports URL templates and server satisfies RESTful requests

b) Test Method: Verify that the server provides a Service Metadata document that includes complete ResourceURL information with resourceType=tile on Layer section if tiles of this layer are able for RESTful. Verify that the template processor in the client is able to convert the URL template in a correct URL to a tile and the server is able to respond RESTful requests.

c) Reference: Subclause 10.2.1

d) Test Type: Capability

A.3.5.3 Tile TileMatrixSet

a) Test purpose: Verify that when a request contains a TileMatrixSet incorrect value, then the server throws an exception.

b) Test method: When a request contains a TileMatrixSet value that is not advertised for this Layer in the ServiceMetadata document, then the server throws an exception (code= InvalidParameterValue) in Procedure Oriented Architectural Style or a HTTP 404 File not found in Resource Oriented Architectural Style.

c) Reference: Subclause 7.2.2.2 and Table 26

d) Test type: Capability

A.3.5.4 Tile TileMatrix

a) Test purpose: Verify that when a request contains a TileMatrix incorrect value, then the server throws an exception.

b) Test method: When a request contains a TileMatrix value that is not advertised for this TileMatrixSet in the ServiceMetadata document, then the server throws an exception (code= InvalidParameterValue) in Procedure Oriented Architectural Style or a HTTP 404 File not found in Resource Oriented Architectural Style.

Copyright © 2010 Open Geospatial Consortium Inc.

c) Reference: Subclause 7.2.2.2 and Table 26

d) Test type: Capability

A.3.5.5 Tile TileRow and TileCol

a) Test purpose: Verify that when a request contains a TileRow or TileCol incorrect, then the server throws an exception.

b) Test method: When a request contains a TileRow or TileCol greater or equal to MatrixHeight and MatrixWidth respectively for the selected TileMatrix, then the server throws an exception (code= TileOutOfRange) in Procedure Oriented Architectural Style or a HTTP 404 File not found in Resource Oriented Architectural Style.

c) Reference: Subclause 7.2.2.2 and Table 26

d) Test type: Capability

A.3.5.6 Tile incorrect Sytle

a) Test purpose: Verify that when a request contains a Style incorrect value, then the server throws an exception.

b) Test method: When a request contains a Style value that is not advertised for this Layer in the ServiceMetadata document, then the server throws an exception (code= InvalidParameterValue) in Procedure Oriented Architectural Style or a HTTP 404 File not found in Resource Oriented Architectural Style.

c) Reference: Subclause 7.2.2.2 and Table 26

d) Test type: Capability

A.3.5.7 Tile incorrect dimension value

a) Test purpose: Verify that when a request contains an incorrect dimension value, then the server throws an exception.

b) Test method: When a request contains a dimension value that is not advertised for this Layer in the ServiceMetadata document, then the server throws an exception (code= InvalidParameterValue) in Procedure Oriented Architectural Style or a HTTP 404 File not found in Resource Oriented Architectural Style.

c) Reference: Subclause 7.2.2.2 and Table 26

d) Test type: Capability

A.3.5.8 Tile dimension default and current

a) Test purpose: Verify that the server supports 'default' and 'current'.

 Copyright © 2010 Open Geospatial Consortium Inc.

b) Test method: When the ServiceMetadata document advertises a default value or the current value support, request that uses the 'default' and 'current' keywords returns a correct answer.

c) Reference: Subclause 7.1.1 Table 9

d) Test type: Capability

A.3.5.9 GetTile incorrect Format

a) Test purpose: Verify that when a request contains a Format incorrect value, then the server throws an exception.

b) Test method: When a request contains a Format value that is not advertised for this Layer in the ServiceMetadata document, then the server throws an exception (code= InvalidParameterValue) in Procedure Oriented Architectural Style

c) Reference: Subclause 7.2.2.2 and Table 26

d) Test type: Capability

A.3.5.10 Tile correct Format

a) Test purpose: Verify that for each GetTile format, when the Format parameter is set to that format or URLtemplate having a format parameter, the MIME type of the response matches that format.

b) Test method: Pass if for each GetTile format, when the Format parameter is set to that format, the MIME type of the response matches that format in Procedure Oriented Architectural Style or the URL template format of the response matches that format in Resource Oriented Architectural Style

c) Reference: Subclause 7.2.1 and 10.2.1

d) Test type: Capability

A.3.5.11 GetTile size

a) Test purpose: Verify that the returned tile has the correct size.

b) Test method: Send a correct request for each TileMatrix in the TileMatrixSet of a Layer and the test passes if width and height of the returned image are equal to the advertised values in TileWidth and TileHeight in the ServiceMetadata.

c) Reference: Subclause 7.2.1

d) Test type: Capability

A.3.5.12 GetTile transparent color

a) Test purpose: Verify that the returned tile has transparent color for NODATA values.

Copyright © 2010 Open Geospatial Consortium Inc.

b) Test method: Send a correct request for a Layer in a format that supports transparency and in a tile where NODATA values are expected and test for transparent color there.

c) Reference: Subclause 7.2.1

d) Test type: Capability

A.3.6 FeatureInfo request test module

In order to pass this test module test A.3.5.1 to A.3.5.9 has to be also passed.

A.3.6.1 GetTileFeatureInfo on non-queryable layer

a) Test purpose: Verify that when a request to a non-queryable layer, then the server throws an exception.

b) Test method: When a GetFeatureInfo is requested on a Layer that not advertises any InfoFormat in the ServiceMetadata document, then the server throws an exception (code=OperationNotSupported) in Procedure Oriented Architectural Style

c) Reference: Subclause 7.3.2.2

d) Test type: Capability

A.3.5.2 FeatureInfo ResourceURL template

a) Test Purpose: Verify that a client supports URL templates and server satisfies RESTful requests

b) Test Method: Verify that the server provides a ServiceMetadata document that includes complete ResourceURL information with resourceType=FeatureInfo on Layer section if FeatureInfos of this layer are able for RESTful. Verify that the template processor in the client is able to convert the URL template in a correct URL to a FeatureInfo and the server is able to respond RESTful requests.

c) Reference: Subclause 10.2.1

d) Test Type: Capability

A.3.6.2 GetFeatureInfo incorrect InfoFormat

a) Test purpose: Verify that when a request contains an InfoFormat incorrect value, then the server throws an exception.

b) Test method: When a request contains an InfoFormat value that is not advertised for this Layer in the ServiceMetadata document, then the server throws an exception (code= InvalidParameterValue) in Procedure Oriented Architectural Style.

c) Reference: Subclause 7.3.2.2

d) Test type: Capability

 Copyright © 2010 Open Geospatial Consortium Inc.

A.3.6.3 FeatureInfo correct InfoFormat

a) Test purpose: Verify that for each GetTile format, when the InfoFormat parameter is set to that format or URLtemplate having a format parameter, the MIME type of the response matches that format.

b) Test method: Pass if for each GetTile format, when the InfoFormat parameter is set to that format, the MIME type of the response matches that format or the URL template format of the response matches that format in Resource Oriented Architectural Style

c) Reference: Subclause 7.3.1 and 10.3.1

d) Test type: Capability

A.3.6.4 FeatureInfo J and I

a) Test purpose: Verify that when a request contains an I or J incorrect, then the server throws an exception.

b) Test method: When a request contains an I or J greater or equal to TileHeight and TileWidth respectively for the selected TileMatrix, then the server throws an exception (code=PointIJOutOfRange) in Procedure Oriented Architectural Style or a HTTP 404 File not found in Resource Oriented Architectural Style.

c) Reference: Subclause 7.3.2.2

d) Test type: Capability

Copyright © 2010 Open Geospatial Consortium Inc.

Annex B
(normative)

XML Schema Documents

In addition to this document, this standard includes several normative XML Schema Documents. These XML Schema Documents may be bundled in a zip file with the present document. After OGC acceptance of a Version 1.0.0 of this standard, these XML Schema Documents will also be posted online at the URL http://schemas.opengis.net/wmts/1.0.0. In the event of a discrepancy between the bundled and online versions of the XML Schema Documents, the online files SHALL be considered authoritative.

The WMTS abilities specified in this document use one specified XML Schema Documents included in the zip file with this document. These XML Schema Documents combine the XML schema fragments listed in various subclauses of this document, eliminating duplications. These XML Schema Documents are named:

wmtsGetCapabilities_request.xsd

wmtsGetCapabilities_response.xsd

wmtsGetTile_request.xsd

wmtsGetFeatureInfo_request.xsd

wmtsGetFeatureInfo_response.xsd

wmtsPayload_response.xsd

In addition, the following XML Schema Documents imported by WSDL documents are used in the annex F and are also included:

wmts.xsd

wmtsKVP.xsd

These XML Schema Documents use and build on the OWS common XML Schema Documents specified OWS Common [OGC 06-121r3], named:

ows19115subset.xsd

owsCommon.xsd

owsDataIdentification.xsd

owsExceptionReport.xsd

owsGetCapabilities.xsd

owsOperationsMetadata.xsd

owsServiceIdentification.xsd

 Copyright © 2010 Open Geospatial Consortium Inc.

owsServiceProvider.xsd

All these XML Schema Documents contain documentation of the meaning of each element and attribute, and this documentation SHALL be considered normative as specified in subclause 11.6.3 of OWS Common [OGC 06-121r3].

Also complete examples for KVP, SOAP and RESTful can be found on the zip file and on the portal. Some fragments of these examples are shown throughout this document.

Copyright © 2010 Open Geospatial Consortium Inc.

Annex C
(informative)

UML model

C.1 Introduction

This annex provides a UML model of the WMTS interface, using the OGC/ISO profile of UML summarized in subclause 5.2 of [06-121r3].

Figure C.1 is a simple UML diagram summarizing the WMTS interface. This class diagram shows that the WMTS class inherits the getCapabilities operation from the OGCWebService interface class, and adds the getTile and getFeatureInfo operations. (The capitalization of names uses the OGC/ISO profile of UML.)

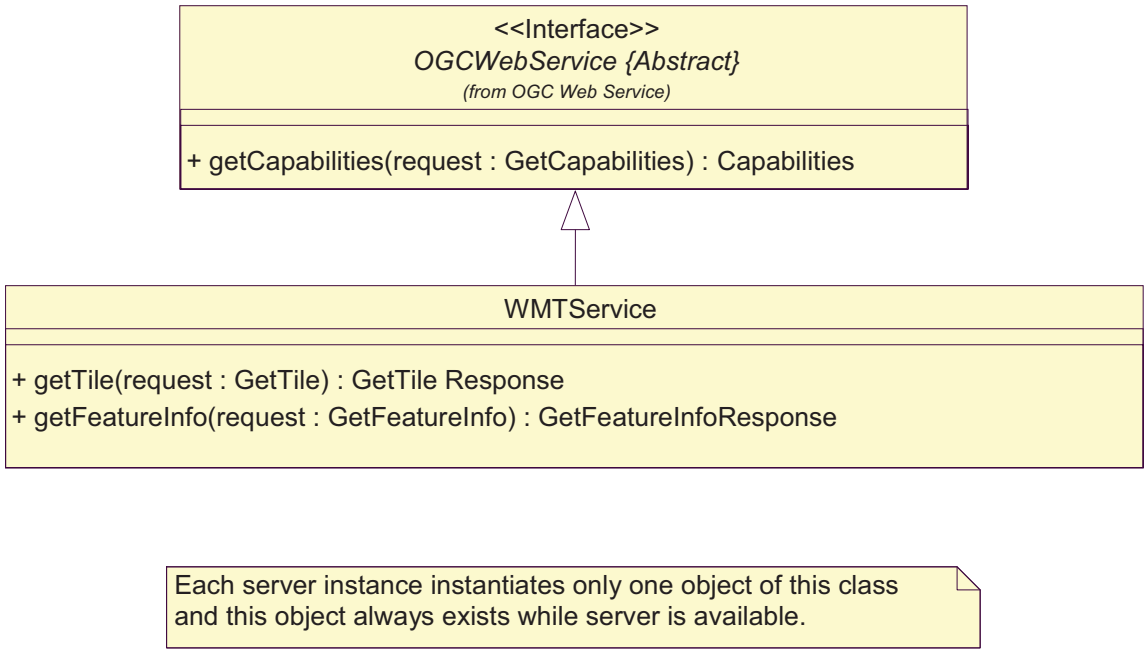


Figure C.1 — WMTS interface UML diagram

Each of the three operations uses a request and a response data type, each of which is defined by one or more additional UML classes. The following subclauses provide a more complete UML model of the WMTS interface, adding UML classes defining the operation request and response data types.

Copyright © 2010 Open Geospatial Consortium Inc.

C.2 UML packages

The WMTS interface UML model is organized in six packages that will be described in the following subclauses. These six WMTS-specific packages make use of six non-WMTS-specific packages, named OWS Web Service, OWS Operations Metadata, OWS Service Identification, OWS Service Provider, and ISO 19115 Subset. This package diagram shows the dependencies among the various packages shown.

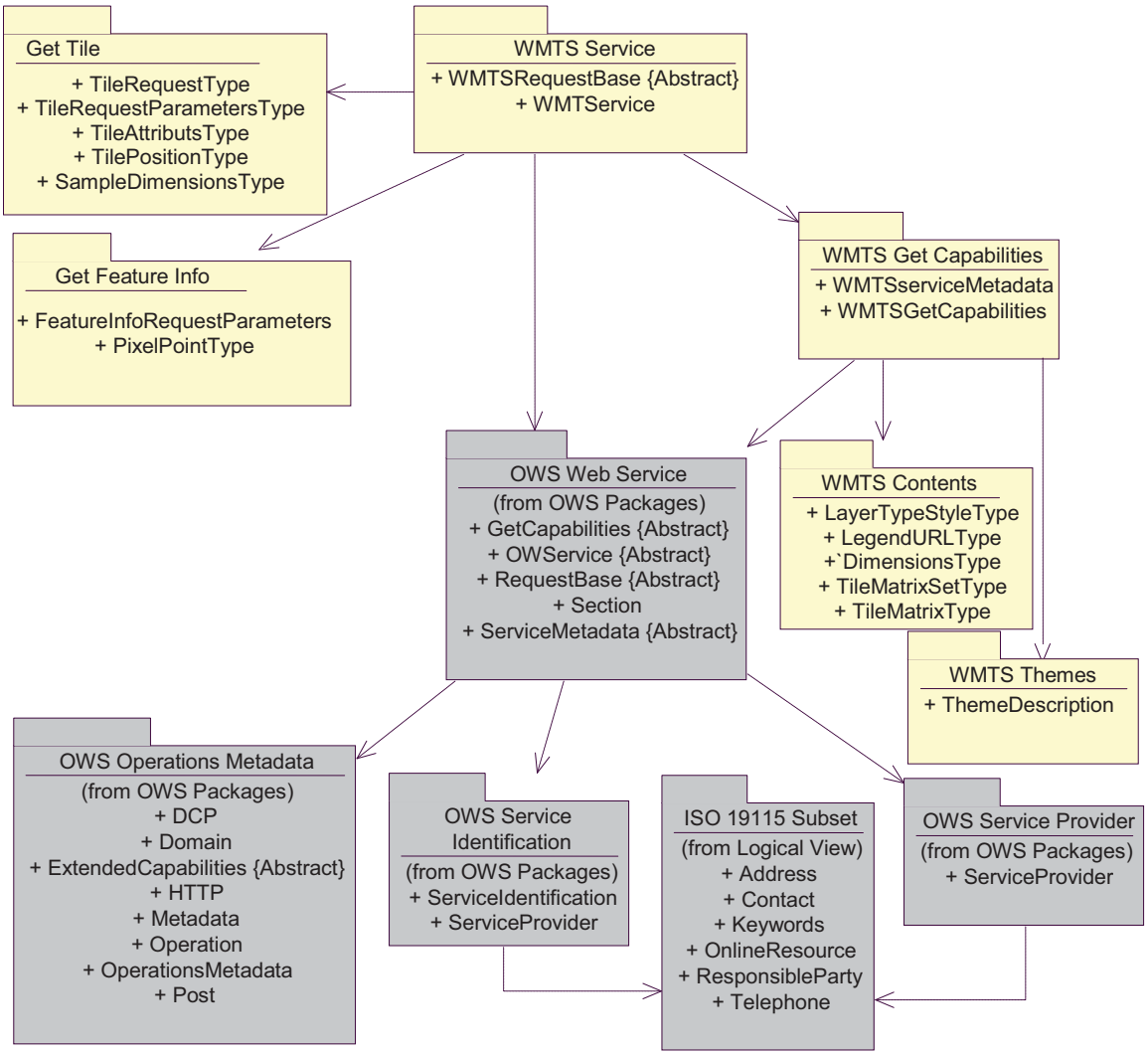


Figure C.2 — WMTS interface package diagram

Each of the three WMTS-specific packages shown in Figure C.2 is described in the following subclauses. The OWS Web Service, OWS Operations Metadata, OWS Service Identification, OWS Service Provider, and ISO 19115 Subset packages are described in the Annex C of OWS Common [OGC 06-121r3].

Copyright © 2010 Open Geospatial Consortium Inc.

C.3 WMTS Service package

The WMTS Service package is shown in the class diagram in Figure C.3. This diagram does not show the classes used by the WMTS operation requests and responses, which are shown (with part of this package) in the WMTS Get Capabilities, Get Tile and Get Feature Info packages. This diagram also shows RequestBase used classes from the OWS Web Service package, which is common to all OGC Web Services, plus one used class from the WMTS package.

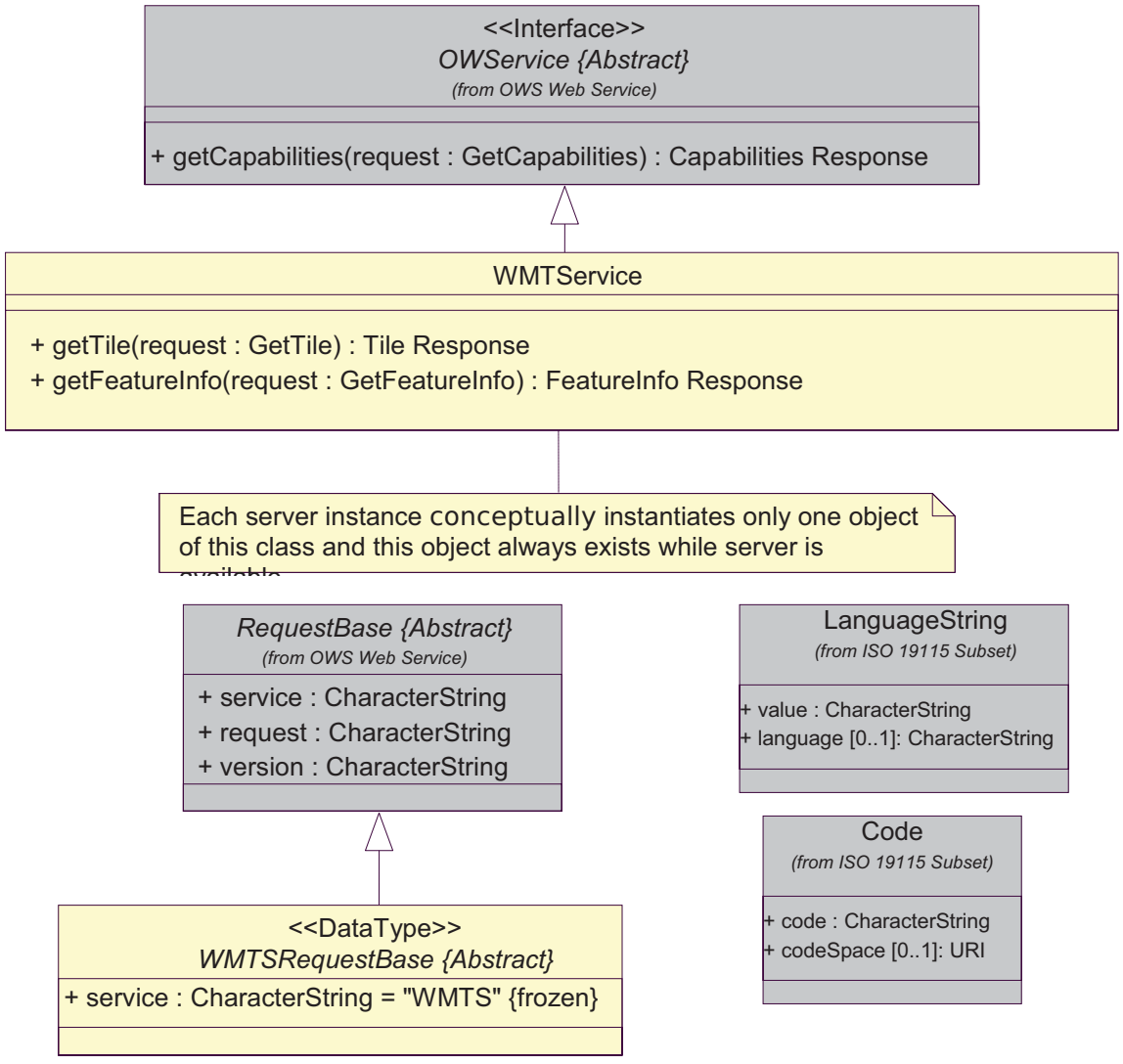


Figure C.3 — WMTS Service package class diagram

Copyright © 2010 Open Geospatial Consortium Inc.

C.4 WMTS Get Capabilities, Contents and Themes packages

The WMTS Get Capabilities package is shown in the class diagram in Figure C.4, C.5 and C.6. This diagram also shows several classes from the OWS package. The classes introduced by this package are further defined by Table 2 through Table 16 in this document.

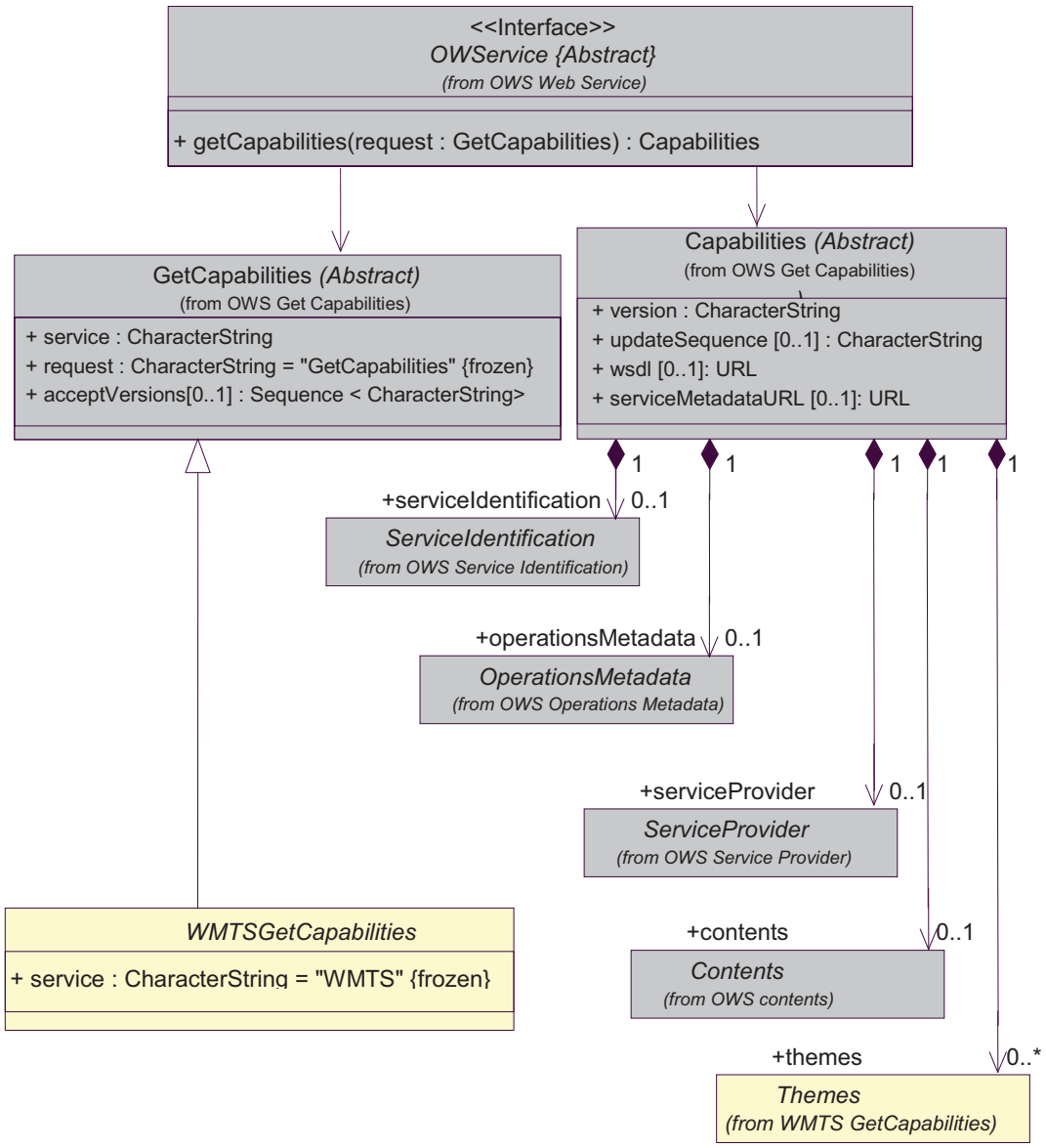


Figure C.4 — Get Capabilities package class diagram, part 1

Copyright © 2010 Open Geospatial Consortium Inc.

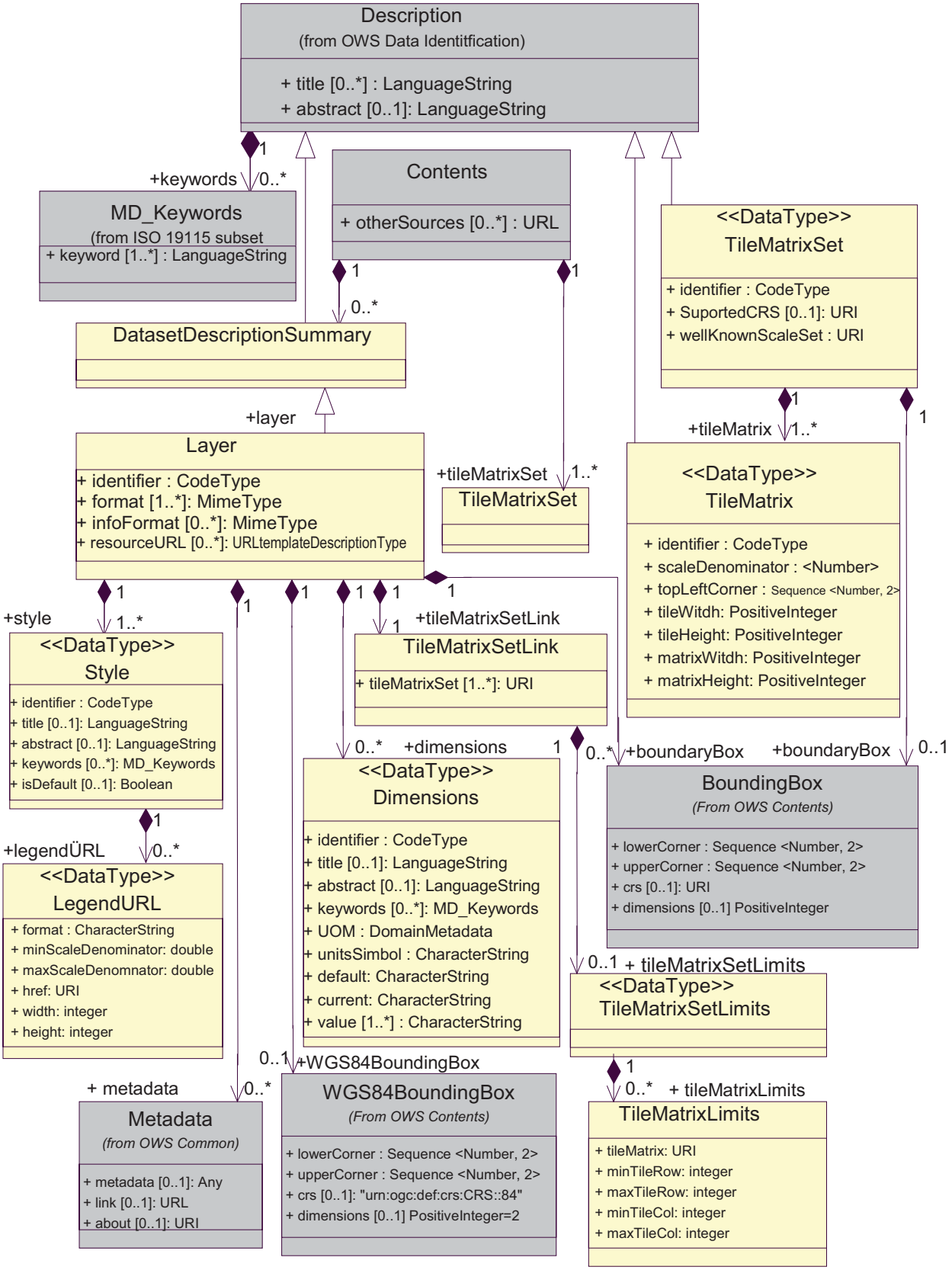


Figure C.5 — Get Capabilities package class diagram, part 2

Copyright © 2010 Open Geospatial Consortium Inc.

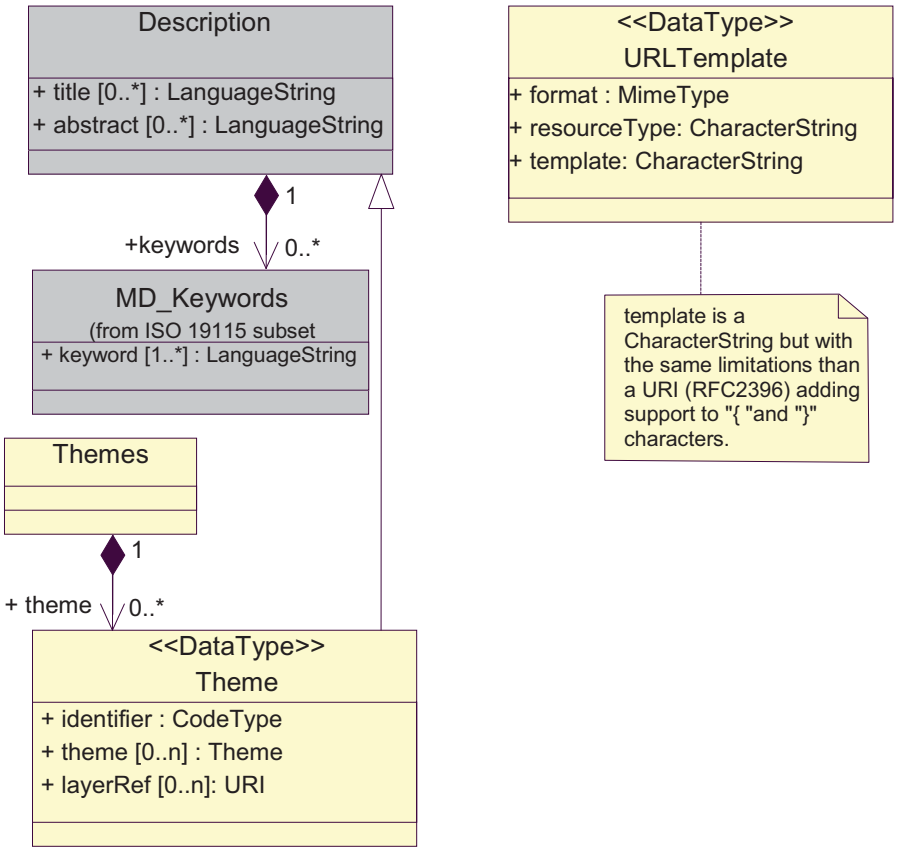


Figure C.6 — Get Capabilities package class diagram, part 3

Copyright © 2010 Open Geospatial Consortium Inc.

C.5 WMTS Get Tile package

The WMTS Get Tile package is shown in the class diagram in Figure C.7. This diagram also shows several classes from the OWS package. The classes introduced by this package are further defined by Table 22 in this document.

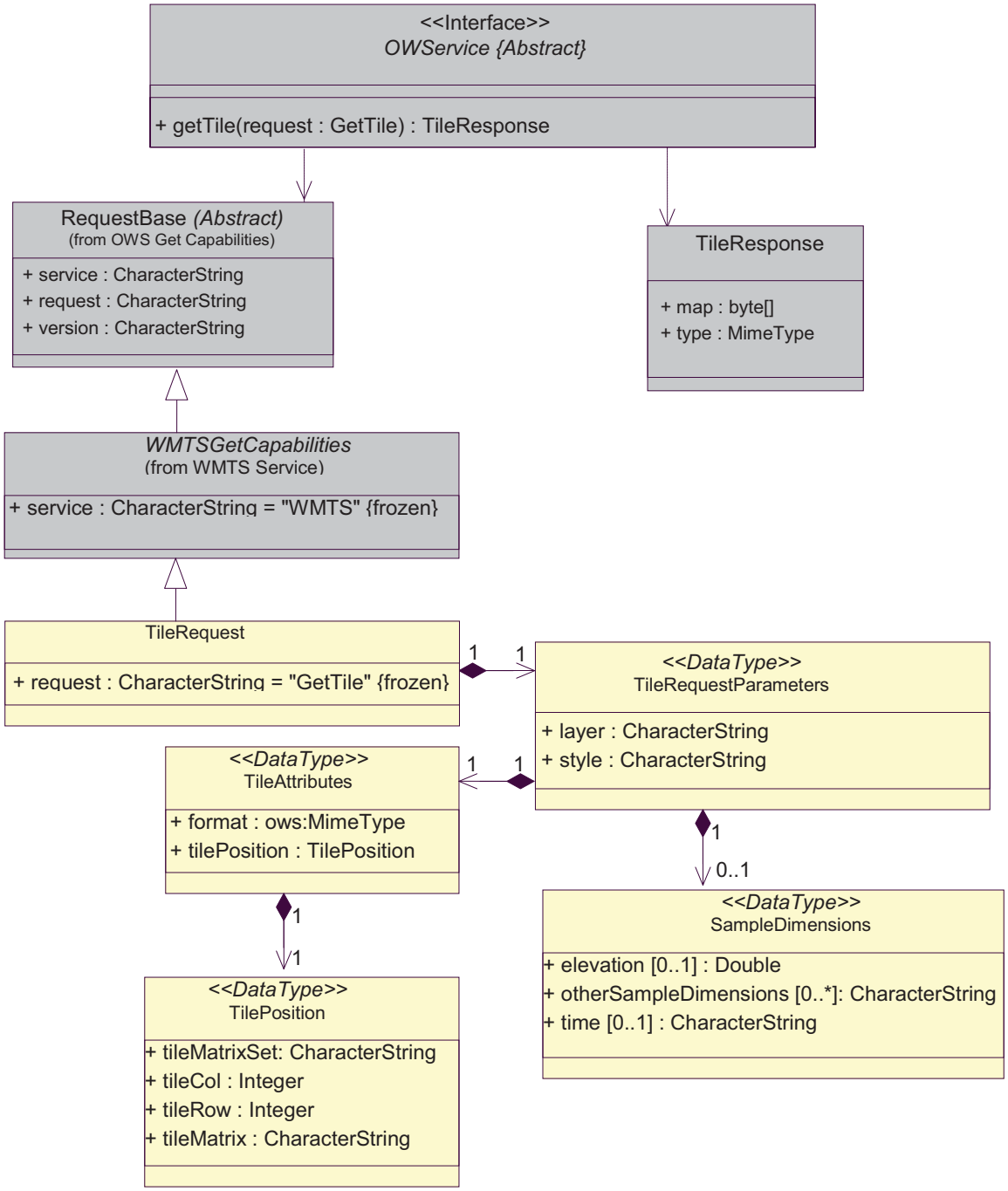


Figure C.7 — Get Tile package class diagram

Copyright © 2010 Open Geospatial Consortium Inc.

Table C.1 — Mapping of UML TileRequest attributes and HTTP GetTile request parameters

UML attribute	HTTP request parameters
service	`Service`
request	`Request`
version	`Version`
layer	`Layer`
style	`Style`
tileAttributes	`TileMatrixSet, TileMatrix, TileRow, TileCol, Format`
sampleDimensions	`Time, Elevation`, *SampleDimensions*

Copyright © 2010 Open Geospatial Consortium Inc.

C.6 WMTS Get Feature Info package

The WMTS Get Feature Info package is shown in the class diagram in Figure C.8. This diagram also shows several classes from the OWS and WMTS packages. The classes introduced by this package are further defined by Table 25 in this document.

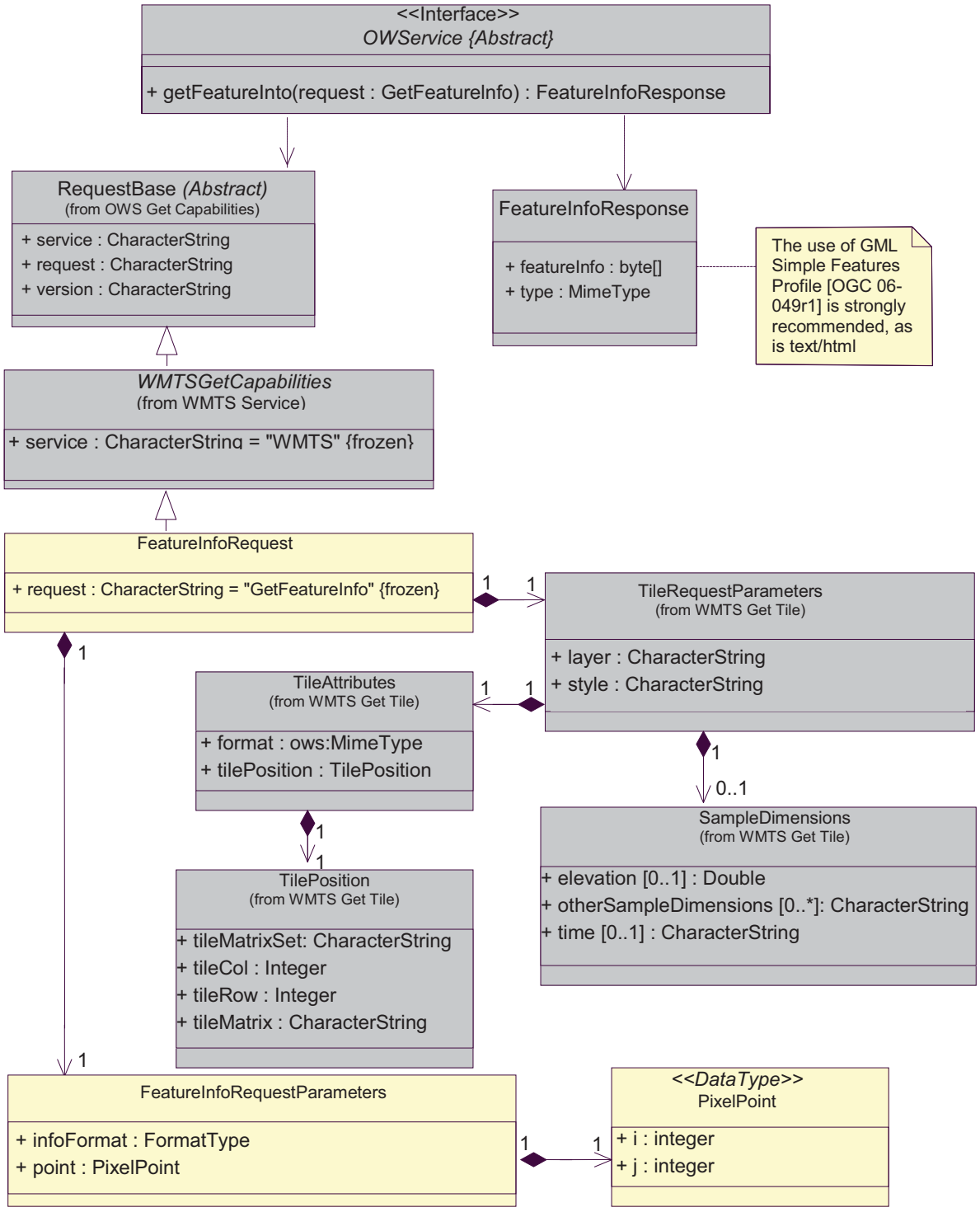


Figure C.8 — Get Feature Info package class diagram

Copyright © 2010 Open Geospatial Consortium Inc.

Table C.2 — Mapping of UML FeatureInfoRequest attributes and HTTP GetFeatureInfo request parameters

UML attribute	HTTP request parameters
service	`Service`
request	`Request`
version	`Version`
layer	`Layer`
style	`Style`
tileAttributes	`TileMatrixSet, TileMatrix, TileRow, TileCol, Format`
sampleDimensions	`Time, Elevation`, *SampleDimensions*
infoFormat	`InfoFormat`
point	`J, I`

Copyright © 2010 Open Geospatial Consortium Inc.

Annex D
(informative)

Example XML documents

D.1 Introduction

This annex provides more example XML documents than given in the body of this document. Particularly it includes an XML example for RESTful approach.

D.2 ServiceMetadata response example

This is a complete example of ServiceMetadata response with RESTful and KVP support.

```
<?xml version="1.0" encoding="UTF-8"?>
<Capabilities xmlns="http://www.opengis.net/wmts/1.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml"
xsi:schemaLocation="http://www.opengis.net/wmts/1.0
http://schemas.opengis.net/wmts/1.0.0/wmtsGetCapabilities_response.xsd"
version="1.0.0">
   <ows:ServiceIdentification>
      <ows:Title>Web Map Tile Service</ows:Title>
      <ows:Abstract>Service that contrains the map
            access interface to some TileMatrixSets</ows:Abstract>
      <ows:Keywords>
         <ows:Keyword>tile</ows:Keyword>
         <ows:Keyword>tile matrix set</ows:Keyword>
         <ows:Keyword>map</ows:Keyword>
      </ows:Keywords>
      <ows:ServiceType>OGC WMTS</ows:ServiceType>
      <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
      <ows:Fees>none</ows:Fees>
      <ows:AccessConstraints>none</ows:AccessConstraints>
   </ows:ServiceIdentification>
   <ows:ServiceProvider>
      <ows:ProviderName>MiraMon</ows:ProviderName>
      <ows:ProviderSite xlink:href="http://www.creaf.uab.cat/miramon"/>
      <ows:ServiceContact>
         <ows:IndividualName>Joan Maso Pau</ows:IndividualName>
         <ows:PositionName>Senior Software Engineer</ows:PositionName>
         <ows:ContactInfo>
            <ows:Phone>
               <ows:Voice>+34 93 581 1312</ows:Voice>
               <ows:Facsimile>+34 93 581 4151</ows:Facsimile>
            </ows:Phone>
            <ows:Address>
```


Copyright © 2010 Open Geospatial Consortium Inc.

```
                <ows:DeliveryPoint>Fac Ciencies UAB</ows:DeliveryPoint>
                <ows:City>Bellaterra</ows:City>
                <ows:AdministrativeArea>Barcelona
                </ows:AdministrativeArea>
                <ows:PostalCode>08193</ows:PostalCode>
                <ows:Country>Spain</ows:Country>
                <ows:ElectronicMailAddress>joan.maso@uab.cat
                </ows:ElectronicMailAddress>
            </ows:Address>
        </ows:ContactInfo>
    </ows:ServiceContact>
</ows:ServiceProvider>
<ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get xlink:href=
            "http://www.maps.bob/cgi-bin/MiraMon5_0.cgi?">
                    <ows:Constraint name="GetEncoding">
                        <ows:AllowedValues>
                            <ows:Value>KVP</ows:Value>
                        </ows:AllowedValues>
                    </ows:Constraint>
                </ows:Get>
            </ows:HTTP>
        </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetTile">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get xlink:href=
            "http://www.maps.bob/cgi-bin/MiraMon5_0.cgi?">
                    <ows:Constraint name="GetEncoding">
                        <ows:AllowedValues>
                            <ows:Value>KVP</ows:Value>
                        </ows:AllowedValues>
                    </ows:Constraint>
                </ows:Get>
            </ows:HTTP>
        </ows:DCP>
    </ows:Operation>
</ows:OperationsMetadata>
<Contents>
    <Layer>
        <ows:Title>Blue Marble Next Generation</ows:Title>
        <ows:Abstract>Blue Marble Next Generation NASA Product
        </ows:Abstract>
        <ows:WGS84BoundingBox>
            <ows:LowerCorner>-180 -90</ows:LowerCorner>
            <ows:UpperCorner>180 90</ows:UpperCorner>
        </ows:WGS84BoundingBox>
        <ows:Identifier>BlueMarbleNextGeneration</ows:Identifier>
        <Style isDefault="true">
            <ows:Identifier>Default</ows:Identifier>
        </Style>
        <Format>image/jpeg</Format>
        <TileMatrixSetLink>
            <TileMatrixSet>BigWorldPixel</TileMatrixSet>
```

Copyright © 2010 Open Geospatial Consortium Inc.

```
        </TileMatrixSetLink>
        <ResourceURL format="image/png" resourceType="tile"
            template="http://www.maps.bob/wmts/BlueMarbleNextGeneration/
            default/BigWorldPixel/{TileMatrix}/{TileRow}/{TileCol}.png"/>
    </Layer>
    <TileMatrixSet>
        <ows:Identifier>BigWorldPixel</ows:Identifier>
        <ows:SupportedCRS>urn:ogc:def:crs:OGC:1.3:CRS84
        </ows:SupportedCRS>
        <WellKnownScaleSet>urn:ogc:def:wkss:OGC:1.0:GlobalCRS84Pixel
        </WellKnownScaleSet>
        <TileMatrix>
            <ows:Identifier>10000m</ows:Identifier>
            <ScaleDenominator>33130800.83133142</ScaleDenominator>
            <TopLeftCorner>-180 90</TopLeftCorner>
            <TileWidth>640</TileWidth>
            <TileHeight>480</TileHeight>
            <MatrixWidth>7</MatrixWidth>
            <MatrixHeight>5</MatrixHeight>
        </TileMatrix>
        <TileMatrix>
            <ows:Identifier>20000m</ows:Identifier>
            <ScaleDenominator>66261601.66266284</ScaleDenominator>
            <TopLeftCorner>-180 90</TopLeftCorner>
            <TileWidth>640</TileWidth>
            <TileHeight>480</TileHeight>
            <MatrixWidth>4</MatrixWidth>
            <MatrixHeight>3</MatrixHeight>
        </TileMatrix>
        <TileMatrix>
            <ows:Identifier>40000m</ows:Identifier>
            <ScaleDenominator>132523203.3253257</ScaleDenominator>
            <TopLeftCorner>-180 90</TopLeftCorner>
            <TileWidth>640</TileWidth>
            <TileHeight>480</TileHeight>
            <MatrixWidth>2</MatrixWidth>
            <MatrixHeight>2</MatrixHeight>
        </TileMatrix>
        <TileMatrix>
            <ows:Identifier>60000m</ows:Identifier>
            <ScaleDenominator>198784804.9879885</ScaleDenominator>
            <TopLeftCorner>-180 90</TopLeftCorner>
            <TileWidth>640</TileWidth>
            <TileHeight>480</TileHeight>
            <MatrixWidth>1</MatrixWidth>
            <MatrixHeight>1</MatrixHeight>
        </TileMatrix>
        <TileMatrix>
            <ows:Identifier>120000m</ows:Identifier>
            <ScaleDenominator>397569609.9759771</ScaleDenominator>
            <TopLeftCorner>-180 90</TopLeftCorner>
            <TileWidth>640</TileWidth>
            <TileHeight>480</TileHeight>
            <MatrixWidth>1</MatrixWidth>
            <MatrixHeight>1</MatrixHeight>
        </TileMatrix>
        <TileMatrix>
            <ows:Identifier>240000m</ows:Identifier>
```

 Copyright © 2010 Open Geospatial Consortium Inc.

```
            <ScaleDenominator>795139219.9519541</ScaleDenominator>
            <TopLeftCorner>-180 90</TopLeftCorner>
            <TileWidth>640</TileWidth>
            <TileHeight>480</TileHeight>
            <MatrixWidth>1</MatrixWidth>
            <MatrixHeight>1</MatrixHeight>
         </TileMatrix>
      </TileMatrixSet>
   </Contents>
   <ServiceMetadataURL
         xlink:href="http://www.maps.bob/wmts/1.0.0/WMTSCapabilities.xml"/>
</Capabilities>
```

Copyright © 2010 Open Geospatial Consortium Inc.

Annex E
(informative)

Well-known scale sets

The following well-known scale sets are defined in this standard. To be conformant to these well-known scale sets, a WMTS server SHALL allow responses from the largest scale denominator on the following tables and all intermediate scale denominators down to the most detailed scale resolution of that data; it is therefore not required to support the smallest scale denominators in order to be conformant to a well-known scale set. Cell sizes (pixel size in terrain units) have been calculated assuming 0.28 mm pixel size and the WGS84 equatorial earth diameter.

URN identifiers for each well-known scale set follow the OGC 07-092r3 best practices document.

E.1 GlobalCRS84Scale (urn:ogc:def:wkss:OGC:1.0:GlobalCRS84Scale)

This well-known scale set has been defined for global cartographic products. Rounded scales have been chosen for intuitive cartographic representation of vector data. Scale denominator is only accurate near the equator.

Table E.1 — Definition of Well-known scale set GlobalCRS84Scale

CRS	Scale Denominator	Pixel Size (degrees)
urn:ogc:def:crs:OGC:1.3:CRS84	$500\ 10^6$	1.25764139776733
	$250\ 10^6$	0.628820698883665
	$100\ 10^6$	0.251528279553466
	$50\ 10^6$	0.125764139776733
	$25\ 10^6$	$6.28820698883665\ 10^{-2}$
	$10\ 10^6$	$2.51528279553466\ 10^{-2}$
	$5\ 10^6$	$1.25764139776733\ 10^{-2}$
	$2.5\ 10^6$	$6.28820698883665\ 10^{-3}$
	$1\ 10^6$	$2.51528279553466\ 10^{-3}$
	$500\ 10^3$	$1.25764139776733\ 10^{-3}$
	$250\ 10^3$	$6.28820698883665\ 10^{-4}$
	$100\ 10^3$	$2.51528279553466\ 10^{-4}$
	$50\ 10^3$	$1.25764139776733\ 10^{-4}$
	$25\ 10^3$	$6.28820698883665\ 10^{-5}$
	$10\ 10^3$	$2.51528279553466\ 10^{-5}$
	$5\ 10^3$	$1.25764139776733\ 10^{-5}$
	$2.5\ 10^3$	$6.28820698883665\ 10^{-6}$

Copyright © 2010 Open Geospatial Consortium Inc.

CRS	Scale Denominator	Pixel Size (degrees)
	$1\ 10^{3}$	$2.51528279553466\ 10^{-6}$
	500	$1.25764139776733\ 10^{-6}$
	250	$6.28820698883665\ 10^{-7}$
	100	$2.51528279553466\ 10^{-7}$

E.2 GlobalCRS84Pixel (urn:ogc:def:wkss:OGC:1.0:GlobalCRS84Pixel)

This well-known scale set has been defined for global cartographic products. Rounded pixel sizes have been chosen for intuitive cartographic representation of raster data. Some values have been chosen to coincide with original pixel size of commonly used global products like STRM (1" and 3"), GTOPO (30") or ETOPO (2' and 5'). Scale denominator and approximated pixel size in meters are only accurate near the equator.

Table E.2 — Definition of Well-known scale set GlobalCRS84Pixel

CRS	Scale Denominator	Pixel Size (degrees)	Approx. Pixel Size (m)
urn:ogc:def:crs:OGC:1.3:CRS84	795139219.9519541	2	240000
	397569609.9759771	1	120000
	198784804.9879885	0.5 (30')	60000
	132523203.3253257	0.333333333333333 (20')	40000
	66261601.66266284	0.166666666666667 (10')	20000
	33130800.83133142	$8.333333333333333\ 10^{-2}$ (5')	10000
	13252320.33253257	$3.333333333333333\ 10^{-2}$ (2')	4000
	6626160.166266284	$1.666666666666667\ 10^{-2}$ (1')	2000
	3313080.083133142	$8.333333333333333\ 10^{-3}$ (30")	1000
	1656540.041566571	$4.166666666666667\ 10^{-3}$ (15")	500
	552180.0138555236	$1.388888888888889\ 10^{-3}$ (5")	166
	331308.0083133142	$8.333333333333333\ 10^{-4}$ (3")	100
	110436.0027711047	$2.777777777777778\ 10^{-4}$ (1")	33
	55218.00138555237	$1.388888888888889\ 10^{-4}$ (0.5")	16
	33130.80083133142	$8.333333333333333\ 10^{-5}$ (0.3")	10
	11043.60027711047	$2.777777777777778\ 10^{-5}$ (0.1")	3
	3313.080083133142	$8.333333333333333\ 10^{-6}$ (0.03")	1
	1104.360027711047	$2.777777777777778\ 10^{-6}$ (0.01")	0.33

Copyright © 2010 Open Geospatial Consortium Inc.

E.3 GoogleCRS84Quad (urn:ogc:def:wkss:OGC:1.0:GoogleCRS84Quad)

This well-known scale set has been defined to allow quadtree pyramids in CRS84. Level 0 allows representing the whole world in a single 256x256 pixels (where the first 64 and last 64 lines of the tile are left blank). The next level represents the whole world in 2x2 tiles of 256x256 pixels and so on in powers of 2. Scale denominator is only accurate near the equator.

Table E.3 — Definition of Well-known scale set GoogleCRS84Quad

CRS	Scale Denominator	Pixel Size (degrees)
urn:ogc:def:crs:OGC:1.3:CRS84	559082264.0287178	1.40625000000000
	279541132.0143589	0.703125000000000
	139770566.0071794	0.351562500000000
	69885283.00358972	0.175781250000000
	34942641.50179486	$8.78906250000000\ 10^{-2}$
	17471320.75089743	$4.39453125000000\ 10^{-2}$
	8735660.375448715	$2.19726562500000\ 10^{-2}$
	4367830.187724357	$1.09863281250000\ 10^{-2}$
	2183915.093862179	$5.49316406250000\ 10^{-3}$
	1091957.546931089	$2.74658203125000\ 10^{-3}$
	545978.7734655447	$1.37329101562500\ 10^{-3}$
	272989.3867327723	$6.86645507812500\ 10^{-4}$
	136494.6933663862	$3.43322753906250\ 10^{-4}$
	68247.34668319309	$1.71661376953125\ 10^{-4}$
	34123.67334159654	$8.58306884765625\ 10^{-5}$
	17061.83667079827	$4.29153442382812\ 10^{-5}$
	8530.918335399136	$2.14576721191406\ 10^{-5}$
	4265.459167699568	$1.07288360595703\ 10^{-5}$
	2132.729583849784	$5.36441802978516\ 10^{-6}$

Copyright © 2010 Open Geospatial Consortium Inc.

E.4 GoogleMapsCompatible (urn:ogc:def:wkss:OGC:1.0:GoogleMapsCompatible)

This well-known scale set has been defined to be compatible with Google Maps and Microsoft Live Map projections and zoom levels. Level 0 allows representing the whole world in a single 256x256 pixels. The next level represents the whole world in 2x2 tiles of 256x256 pixels and so on in powers of 2. Scale denominator is only accurate near the equator.

Table E.4 — Definition of Well-known scale set GoogleMapsCompatible

CRS	Zoom level name	Scale Denominator	Pixel Size (m)
urn:ogc:def:crs:EPSG:6.18:3:3857 WGS 84 / Pseudo-Mercator http://www.epsg-registry.org/export.htm?gml=urn:ogc:def:crs:EPSG::3857	0	559082264.0287178	156543.0339280410
	1	279541132.0143589	78271.51696402048
	2	139770566.0071794	39135.75848201023
	3	69885283.00358972	19567.87924100512
	4	34942641.50179486	9783.939620502561
	5	17471320.75089743	4891.969810251280
	6	8735660.375448715	2445.984905125640
	7	4367830.187724357	1222.992452562820
	8	2183915.093862179	611.4962262814100
	9	1091957.546931089	305.7481131407048
	10	545978.7734655447	152.8740565703525
	11	272989.3867327723	76.43702828517624
	12	136494.6933663862	38.21851414258813
	13	68247.34668319309	19.10925707129406
	14	34123.67334159654	9.554628535647032
	15	17061.83667079827	4.777314267823516
	16	8530.918335399136	2.388657133911758
	17	4265.459167699568	1.194328566955879
	18	2132.729583849784	0.5971642834779395

Copyright © 2010 Open Geospatial Consortium Inc.

Annex F
(normative)

WSDL description of the service

This Annex is normative for services that support SOAP messages. It provides an abstract WSDL description for a generic WMTS service and guidance on how to create a concrete WSDL description for a particular WMTS server instance.

A WSDL document is typically used in combination with SOAP encoding but this annex describes WSDL documents that deal with KVP and SOAP encodings of the procedure oriented architectural style. WSDL documents (especially in version 1.1) seem not to be appropriate to describe resource oriented architectural style.

F.1 General

The Web Services Description Language (WSDL) is an XML language for describing the computational characteristics of a web service: interface signatures, protocol bindings and network endpoints. Version 1.1 was authored jointly by Microsoft and IBM and published as a W3C Note in March 2001. Furthermore, WSDL 1.1 is recommended by WS-I Basic Profile 1.2 and the current drafts of WS-I Basic Profile 2.0, which will be used in conjunction with WSDL 1.1. WSDL 2.0 (formerly 1.2 but renamed because of its substantial differences from 1.1) has already been promoted to a W3C Recommendation but is not widely supported yet by standards and tools.

F.2 WSDL Publication

There are many ways to publish a WSDL file for a Web Service instance. The mainstream IT world has established three major ways:

1. The WSDL file can be offered on the web site of the organization that publishes the web service. This approach allows humans to find the WSDL file but is not sufficient for automatic use.

2. The WSDL file can be published through public and private registries. UDDI would be the choice for general Web Services and CSW for OGC Web Services. This approach follows the publish-find-bind pattern and thus allows humans and services to discover the WSDL file in a standardized manner.

3. The web service itself can also publish the WSDL file. AXIS and the .NET frameworks follow the convention of http://url:port/service/xx?WSDL. This is sufficient for a pragmatic approach, but fails for multiple WSDL files describing specific aspects of a Web Service as long as the base URL does not change.

 Copyright © 2010 Open Geospatial Consortium Inc.

Option number two can be combined with option number three by just pointing to the WSDL initially published via the WSDL pattern. Therefore, it is recommended to publish a single WSDL file as described in option three but ideally to publish it through any kind of registry.

Also, within the service metadata document, a <WSDL> element may be used to specify a reference to a WSDL resource. If the service has a SOAP binding, there SHALL be a <WSDL> element. The value of the xlink:href attribute SHALL refer to a web accessible WSDL document. The xlink:role attribute indicates the namespace of the document element and WSDL version (in our examples http://schemas.xmlsoap.org/wsdl/1.0). The xlink:show attribute has the value "none" to indicate that no specific behavior is intended.

This approach enables OGC Web Services with SOAP bindings to be discovered via the GetCapabilities operation and to get additional information through the referenced WSDL file.

F.3 Abstract and concrete WSDL documents

WSDL documents are intended to be modularized through the use of import statements since they are structured in an abstract and concrete service instance part. These mechanisms permit the separation of service-specific elements from shared interface definitions; such an authoring style is recommended in the WSDL specifications, and it is advocated here. In practice, this separation means that the complete OGC service will always be described by exactly one top-level WSDL. This top-level WSDL file may import a set of WSDL files for specific parts, for instance a WSDL for the abstract part and a WSDL describing only the concrete service instance part of the service.

A concrete WSDL 1.1 document need to describe five main parts: types, messages, portTypes, bindings and services. The abstract WSDL document describes types, messages and portTypes in a generic way and can be imported in the concrete WSDL description of any particular service instance. A concrete WSDL document also has to describe the binding and service parts of the document.

F.4 Abstract WMTS WSDL document

Abstract WSDL document have a modular design that reuses application schemas that are also used by SOAP messages and are mentioned in the Annex B and in some subclauses of this document.

In addition to this document, this standard includes and abstract WSDL document and some XML Schema Documents imported by the abstract WSDL and also mentioned in the Annex B. These XML Schema Documents and the abstract WSDL are bundled in a zip file with the present document. After OGC acceptance of a Version 1.0.0 of this standard, these files will also be posted online at the URL http://schemas.opengis.net/wmts/1.0.0. In the event of a discrepancy between the bundled and online versions of the XML Schema Documents, the online files SHALL be considered authoritative.

Copyright © 2010 Open Geospatial Consortium Inc.

The documents relevant only in WSDL document creation are named:

wmtsAbstract.wsdl

wmts.xsd

wmtsKVP.xsd

Also complete examples can be found on the zip file and on the portal. Some fragments of these examples are shown throughout this document.

This is WSDL schema fragment for the abstract WMTS service illustrates the ability of importing some XDS files:

```
<definitions name="WMTS">

    <types>
        <xsd:schema
        targetNamespace="http://www.opengis.net/wmts_wsdl/1.0">
            <xsd:import namespace="http://www.opengis.net/ows/1.1"
               schemaLocation="../../ows/1.1.0/owsCommon.xsd"/>
            <xsd:import namespace="http://www.opengis.net/wmts/1.0"
               schemaLocation="wmts.xsd"/>
        </xsd:schema>
    </types>

    <message name="GetTileMessage_GET">
        <part name="service" type="wmts:RequestServiceType"/>
        <part name="request" type="wmts:GetTileValueType"/>
        <part name="version" type="wmts:VersionType"/>
        <part name="layer" type="xsd:string"/>
        <part name="style" type="xsd:string"/>
        <part name="format" type="ows:MimeType"/>
        <part name="TileMatrixSet" type="xsd:string"/>
        <part name="TileMatrix" type="xsd:string"/>
        <part name="TileRow" type="xsd:unsignedInt"/>
        <part name="TileCol" type="xsd:unsignedInt"/>
    </message>

    <message name="GetTileMessage_POST">
        <part name="request" element="wmts:GetTile"/>
    </message>
    <message name="GetTileResult_SOAP">
        <part name="body" element="wmts:BinaryPayload" />
    </message>

    <portType name="WMTS_HTTP_Port_GET">
        <operation name="GetTile">
            <input message="tns:GetTileMessage_GET"/>
            <output message="tns:GetTileResult"/>
            <fault name="exception"
                   message="tns:ServiceExceptionMessage"/>
        </operation>
    </portType>
    <portType name="WMTS_HTTP_Port_SOAP">
        <operation name="GetTile">
            <input message="tns:GetTileMessage_POST"/>
            <output message="tns:GetTileResult_SOAP"/>
```

 Copyright © 2010 Open Geospatial Consortium Inc.

```
            <fault name="exception"
                   message="tns:ServiceExceptionMessage"/>
        </operation>
    </portType>

</definitions>
```

F.5 Concrete WMTS WSDL document

A concrete WSDL describes a particular server instance. First, it has to import the abstract WSDL file, then it has to use a <binding> element for each encoding the server supports and, finally, a service element has to be used and it has to contain as port elements as encodings the server supports.

A HTTP GET <binding> element SHALL reference a GET portType from the abstract part and make use of the `<http:binding verb="GET">` binding as described in the WSDL 1.1 specification. The operation element SHALL reference the corresponding operation from the abstract part and use `<http:urlEncoded/>` as input and `<mime />` element as output. The `<http:operation>` element SHALL be used according to the WSDL 1.1 and WS-I Basic.

A SOAP <binding> element SHALL reference a SOAP portType from the abstract part and make use of the `<soap:binding style="document">` binding as described in the WSDL 1.1 specification. The operation element SHALL reference the corresponding operation from the abstract part and use `<soap:body use="literal"/>` as input and output. The `<soap:operation>` element SHALL be used according to the WSDL 1.1 and WS-I Basic Profile 1.2 specifications. The soapAction attribute value SHALL follow the format template:

http://www.opengis.net/{serviceType}/requests#{operationName}

`<port>` elements in the `<service>` element SHALL reference bindings from the binding part. In a GET encoding an `<http:address location="">` element SHALL reference the URL of the service (that has to be the same of the `<ows:Get xlink:href="">` element attribute in the service metadata document). In a SOAP encoding an `<soap:address location="">` element SHALL reference the URL of the service (that has to be the same value of the `<ows:Get xlink:href="">` element attribute in the service metadata document).

F.6 Concrete WMTS WSDL document example

The following example illustrates how a concrete WSDL document imports the abstract WSDL document and adds a particular description of bindings and service for this particular server instance. The example only describes GetTile operation and describes the same server example shown in subclause 7.1.3 of this document:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:wmts_wsdl="http://www.opengis.net/wmts_wsdl/1.0"
             targetNamespace="http://www.opengis.net/wmts_wsdl/1.0">
```

Copyright © 2010 Open Geospatial Consortium Inc.

```
    <!-- import WMTS types, message and portType -->

    <import namespace="http://www.opengis.net/wmts_wsdl/1.0"
                        location="../wmtsAbstract.wsdl"/>

<!--     Bindings    -->

    <!-- HTTP Get KVP  bindings -->

    <binding name="WMTS_HTTP_GET_Binding"
                  type="wmts_wsdl:WMTS_HTTP_Port_GET">

        <http:binding verb="GET"/>

        <operation name="GetTile">
            <http:operation location=""/>
            <input>
                <http:urlEncoded/>
            </input>
            <output>
                  <mime:content type="image/*"/>
            </output>
            <fault name="exception">
                <mime:mimeXML/>
            </fault>
        </operation>

    </binding>

    <!-- HTTP Post SOAP  bindings -->

    <binding name="WMTS_SOAP_Binding"
                   type="wmts_wsdl:WMTS_HTTP_Port_SOAP">

        <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>

        <operation name="GetTile">
            <soap:operation
            soapAction="http://www.opengis.net/wms/requests#GetTile"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                  <soap:body use="literal"/>
            </output>
            <fault name="exception">
                <soap:fault name="exception" use="literal"/>
            </fault>
        </operation>

    </binding>

    <!--    Services     -->
    <service name="WMTS-TiledWorld-UAB-CREAF-MiraMon">

        <port name="WMTS-GET-Port"
                     binding="wmts_wsdl:WMTS_HTTP_GET_Binding">
```

 Copyright © 2010 Open Geospatial Consortium Inc.

```
            <http:address location=
                "http://www.maps.bob/maps.cgi"/>
        </port>

        <port name="WMTS-SOAP-Port"
                    binding="wmts_wsdl:WMTS_SOAP_Binding">
            <soap:address location=
                "http://www.maps.bob/maps.cgi"/>
        </port>

    </service>

</definitions>
```

Copyright © 2010 Open Geospatial Consortium Inc.

Annex H
(informative)

Pseudocode

This informative Annex provides pseudocode that illustrates how to get the tiles that cover a bounding box rectangle and how to get the CRS coordinates that bounds a tile.

H.1 From BBOX to tile indices

The following fragment of pseudocode could be used to convert from a desired bounding box (bBoxMinX, bBoxMinY, bBoxMaxX, bBoxMaxY) in CRS coordinates to a range of tile set indices. This pseudocode uses the same notation that subclause 6.1 uses. In this pseudocode we assume that bBoxMinX, bBoxMinY, bBoxMaxX, bBoxMaxY, tileMatrixMinX, tileMatrixMinY, tileMatrixMinY, tileMatrixMaxY, tileSpanX and tileSpanY are floating point variables (IEEE-754) that has accuracy issues derived from the finite precision of the representation. These accuracy issues could be amplified in a typical floor() rounding down function that could return a value ±1 than that expected. To overcome this issue this code uses a small value (epsilon) added or subtracted in a place that is not affected by CRS coordinate precision.

```
// to compensate for floating point computation inaccuracies
epsilon = 1e-6

tileMinCol = floor((bBoxMinX - tileMatrixMinX) / tileSpanX +
          epsilon)
tileMaxCol = floor((bBoxMaxX - tileMatrixMinX) / tileSpanX -
          epsilon)
tileMinRow = floor((tileMatrixMaxY - bBoxMaxY) / tileSpanY +
          epsilon)
tileMaxRow = floor((tileMatrixMaxY - bBoxMinY) / tileSpanY -
          epsilon)

// to avoid requesting out-of-range tiles
if (tileMinCol < 0) tileMinCol = 0
if (tileMaxCol >= matrixWidth) tileMaxCol = matrixWidth-1
if (tileMinRow < 0) tileMinRow = 0
if (tileMaxRow >= matrixHeight) tileMaxRow = matrixHeight-1
```

To fetch all the tiles that cover this bounding box, a client would scan through tileMinCol to tileMaxCol and tileMinRow to tileMaxRow, all inclusive. A total of (tileMaxCol-tileMinCol+1) × (tileMaxRow- tileMinRow+1) will be fetched.

H.2 From tile indices to BBOX

The following pseudocode could be used to convert from a pair of tile indices (tileCol, tileRow) to the bounding box (in CRS coordinates) of this tile defined by the upper-left corner (leftX, upperY) of the tile:

 Copyright © 2010 Open Geospatial Consortium Inc.

```
leftX = tileCol * tileSpanX + tileMatrixMinX
upperY = tileMatrixMaxY - tileRow * tileSpanY
```

and the lower-right corner (rightX, lowerY) of the tile:

```
rightX = (tileCol+1) * tileSpanX + tileMatrixMinX
lowerY = tileMatrixMaxY – (tileRow+1) * tileSpanY
```

Copyright © 2010 Open Geospatial Consortium Inc.

Bibliography

[1] OGC 05-007r7, OpenGIS® Web Processing Service Implementation Specification version 1.0.0, at http://portal.opengeospatial.org/files/?artifact_id=24151

[2] OGC 06-042, OpenGIS® Web Map Server Implementation Specification version 1.3.0, at http://portal.opengeospatial.org/files/?artifact_id=14416

[3] OGC 06-121r3, OpenGIS® Web Service Common Implementation Specification version 1.1.0 with Corrigendum 1, at http://portal.opengeospatial.org/files/?artifact_id=20040

[4] OGC 07-057r2, OpenGIS® Tiled WMS Discussion Paper version 0.3.0, at http://portal.opengeospatial.org/files/?artifact_id=23206

[5] OGC 07-092r3, Definition identifier URNs in OGC namespace Best Practices version 1.3, at http://portal.opengeospatial.org/files/index.php?artifact_id=30575

[6] OGC 07-146r2 OGC® KML at https://portal.opengeospatial.org/files/?artifact_id=27810

[7] OGC 07-156r1 Integration of Resource-Oriented Architecture Concepts into the OGC Reference Model at http://portal.opengeospatial.org/files/?artifact_id=29634&version=1

[8] OGC 08-142 OWS-Common WSDL Annex

[9] OGC 09-006 OWS-6-DSS Engineering Report – SOAPXML and REST in WMTS at http://portal.opengeospatial.org/files/?artifact_id=31987

[10] OGC 09-144r1, Technical Committee Policies and Procedures: MIME Media Types for GML, Version 1.0, at http://portal.opengeospatial.org/files/?artifact_id=37743

[11] OnEarth WMS tiled extension at http://onearth.jpl.nasa.gov/tiled.html

[12] OpenSearch Specification 1.1 draft 4 http://www.opensearch.org/Specifications/OpenSearch/1.1

[13] OSGeo Tile Map Service Specification version 1.0 at http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification

[14] OSGeo WMS Tile Caching at http://wiki.osgeo.org/wiki/WMS_Tile_Caching

[15] RESTful Web Services. Web Services for the real World. Leonard Richardson and Sam Ruby. O'Reilly Media Inc. 2007.

[16] URI Template, draft-gregorio-uritemplate-03. Network Working Group Internet-Draft. J. Gregorio, Ed. http://tools.ietf.org/html/draft-gregorio-uritemplate-03

 Copyright © 2010 Open Geospatial Consortium Inc.

5. COMBINING JPEG2000 COMPRESSED FORMATS AND OGC STANDARDS FOR FAST AND EASY DISSEMINATION OF LARGE SATELLITE DATA

Aquest capítol és una reproducció de: Masó J., Zabala A. and Pons X. (2010), Combining JPEG2000 Compressed Formats and OGC Standards for Fast and Easy Dissemination of Large Satellite. *Italian Journal of Remote Sensing (Rivista Italiana di Telerilevamento)* 42 (3), 101–114 (en aquest document es referencia com Masó *et al.*, 2010b)

Combining JPEG2000 Compressed Formats and OGC Standards for Fast and Easy Dissemination of Large Satellite Data

Joan Masó[1], Alaiz Zabala[2] and Xavier Pons[2,1]

[1] Center for Ecological Research and Forestry Applications, UAB, 08193 Bellaterra, Barcelona, Spain.
E-mail: joan.maso@uab.cat.
[2] Geography Department of the Autonomous University of Barcelona 08193 Bellaterra, Barcelona, Spain.

Abstract

This article explores ways for disseminating huge dataset imagery compressed in JPEG2000 using OGC standards. Some web service protocols are analyzed and tested (Web Map Service, Web Feature Service, Web Coverage Service, etc) but special attention is given to the new Web Map Tile Service (WMTS) standard that has been recently approved by the OGC Technical Committee. A WMTS server exposes a structure of resolution levels that are cut in small and uniform tiles in a very similar way that JPEG2000 internally does. We discuss how JPEG2000 and WMTS can be combined together and the advantages and disadvantages versus a more common approach based on prerendered classical JPEG or PNG tiles or versus JPIP approaches. Among the different architectures, it is worth mentioning a WMTS server that uses one or some JPEG2000 images internally and serves images to a thin client (a web browser) in a more classical format like JPEG, PNG etc. In this case, WMTS tile matrix set structure can be used to describe the internal structure of the tiled JPEG2000 images to the client, guaranteeing an optimum performance and taking profit of the Internet caching mechanisms.

Keywords: Wavelet, compression, web services, standards.

Combinazione di formati compressi JPEG2000 e OGC Standards per una distribuzione veloce e agevole di grandi immagini satellitari.

Riassunto

Il presente articolo esplora differenti modi per la distribuzione di grandi immagini satellitari compresse in JPEG2000 utilizzando standard OGC. Alcuni protocolli di servizi web sono analizzati e testati (Web Map Service, Web Feature Service, Web Coverage Service, ecc), anche se un'attenzione speciale è rivolta al nuovo standard Web Map Tile Service (WMTS) approvato recentemente dal Comitato Tecnico del OGC. Un server WMTS espone una struttura di livelli di risoluzione che vengono tagliati in piccoli tasselli (tiles) uniformi in un modo molto simile a quello che il JPEG2000 adotta internamente. Discutiamo come il JPEG2000 e il WMTS possono essere combinati insieme, e sui vantaggi e svantaggi rispetto a un approccio più comune basato sui tasselli prerenderizzati del classico JPEG o PNG, o rispetto agli approcci JPIP. Tra le diverse architetture, vale la pena ricordare il

server WMTS che utilizza internamente una o più immagini JPEG2000, e serve immagini ad un thin client (un browser web) in un formato più classico come JPEG, PNG, ecc. In questo caso, la struttura di matrice di tasselli WMTS può essere utilizzata per descrivere la struttura interna delle immagini tassellate JPEG2000 al client, garantendo un ottimo rendimento e sfruttando i meccanismi di memorizzazione nella cache di Internet.
Parole chiave: *Wavelet, compressione, servizi web, standards, norme.*

Introduction

Spatial Data Infrastructures (SDI) are excellent instruments to risk assessment studies and post crisis reaction, among many other applications [Rajabifard et al., 2004]. The main sources of primary information for SDIs are satellite sensors and, eventually, airborne digital cameras that can produce huge amounts of daily raster images. The International Charter Space and Major Disasters makes available these images for free in an event of a crisis. These huge images can be discovered through catalogues and have to be downloaded to the devices of the post crisis reaction teams on the ground. Bandwidth is currently a problem in many parts of the Earth and especially in a crisis situation where some data links could be down and the available communication channels can be overloaded. Image compression techniques are crucial for efficient and fast dissemination of this critical data to the reaction teams that normally use laptop computers and mobile devices. In terms of quality and compression ratio, wavelet compression algorithms, and particularly the JPEG2000 format, have a better performance than other compression techniques [Taubman and Marcellin, 2002]. Another critical factor in a crisis situation is the use of standards that make possible to easily do tasks with any software available, especially in an environment where the operator of the system could be under stress. JPEG2000 has been standardized in the ISO 15444, so interoperability between different software implementations can be guaranteed [ISO/IEC 15444-1, 2000].

JPEG2000 offers superior benefits than JPEG for a wide variety of applications. It supports lossy and lossless compression. It offers improved quality at the same compression ratio (for instance, it does not have JPEG 8x8 block artifacts). It has been designed to facilitate onscreen display of big imagery and has significantly improved bit-stream scalability, which is defined at the image level. However, all these benefits come at the expense of the algorithm complexity. It means that the JPEG2000 compression and decompression times are larger than those for JPEG.

The compression process can be divided in 3 parts: a discrete wavelet transform (DWT), a quantization and an entropy encoding. DWT generates sub-bands of coefficients that are quantized, divided in precincts and collected in rectangular arrays called codeblocks that are entropy coded. A JPEG2000 file consists in a header and a codeblock array. The header contains the necessary information to describe the image and to locate the codeblocks [Christopoulos *et al.*, 2000] (see Fig. 1). By using DWT the compression algorism defines a set of transformation levels. By doing so, the image can be easily decompresses to its original resolution and to a set of power of 2 of the original spatial resolution.

Currently, two main software providers have adopted wavelet based proprietary formats like MrSID (Lizartech) and ECW (ErMapper) that have been successfully used in remote sensing and GIS applications. JPEG2000 format is an open standard that could replace both in the future, especially if can be used in combination with other open standards.

The standards organization Open Geospatial Consortium (OGC) has been successfully proposing several geospatial web services to interoperate with maps, features, coverages and sensors, and they are deeply accepted in the SDI community. However, some of these standards were written before JPEG2000 format started to grow and many still ignore it. Nevertheless, JPEG2000 could be used in several OGC web services to improve performace in the geospatial web.

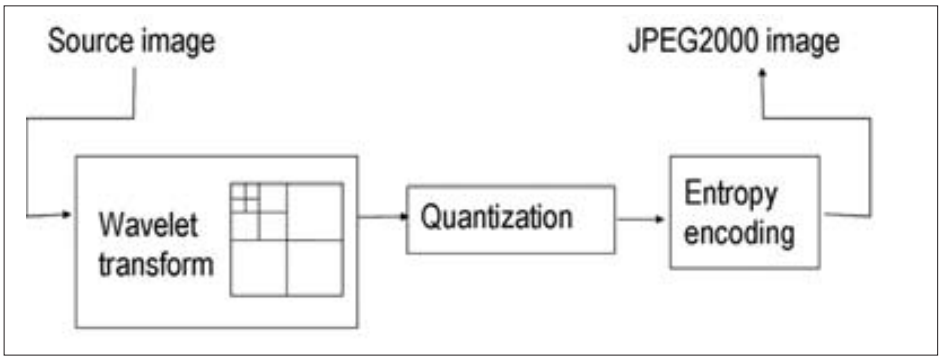


Figure 1 – JPEG2000 encoding process.

Since 2000, OGC has been releasing several standards that are relevant for JPEG2000 data distribution. Particularly WMS specifies how to distribute fragments of maps [de la Beaujardiere, 2004], Web Coverage Service (WCS) specifies how to distribute coverage images (mainly raster files) [Whiteside *et al.*, 2008] and Sensor Observation Service (SOS) specifies how to extract information from a sensor (that could serve images) [Na and Priest, 2007]. Also, OGC has recently approved a new standard called WMTS. This standard targets map services that are expected to have many concurrent requests. In these situations the flexible approach that WMS took and that forces the service to generate maps on the fly can be too time consuming. WMTS proposes a more rigid approach that consists in limiting the zoom levels available to a predefined set of scale levels (called tile matrix set) and, for each scale (called tile matrix), it divides the dataset space in regular and rectangular non-overlapping blocks, called tiles, describing a pyramidal structure [Masó *et al.*, 2009]. WMTS does not tell us how we have to prepare the data on the server but it makes clear that it shall be in a way that a tile extraction can be very fast so the tiles could be prerendered representations of a layer (in a common image format like JPEG) or generated on-the-fly from the original dataset and stored in cache for future requests. We will show how we can optimize the preparation of JPEG2000 images to make the extraction of WMTS tile very fast.
On the other hand, Part 9 of the 15444 ISO standard (JPIP) specifies an interactive protocol that allows transmitting only the needed parts (called codeblocks) of an image that a client is interactively showing on the screen. The server keeps track of all previously transmitted parts and do not send them again, because the client keeps the previous codeblocks in cache. Only never transmitted parts are sent, so that this strategy saves time and bandwidth [ISO/IEC 15444-9, 2003]. This standard has the same purpose than the WMS and WMTS have, but using a totally different strategy that requires a thick client to work over and that is limited to the JPEG2000 format.

This paper exposes the way that huge dataset imagery can be compressed in JPEG2000 and served to the reaction teams in a crisis situation using OGC standards.

Integration in OGC Standards

Using JPEG2000 in SOS

SOS allows requesting data from a specified measure of a specific sensor on a certain time. Particularly, the sensor could be a camera that takes pictures from time to time. The sensor can return observations in JPEG2000 format and other information using the GetObservation request. In this case, the whole JPEG2000 image will be retrieved, since GetObservation has no way to indicate an image resampling or clipping. Nevertheless, there are still advantages when using JPEG2000 formats, especially if the camera is a multispectral sensor, because JPEG2000 can compress between bands taking advantage of their redundancy and generating a smaller file.

Another approach could consist in returning the image using a JPIP service. This is possible because SOS GetObservation request has a parameter called ResponseMode that can be set to out-of-band value. In this case you will not receive the data itself but instructions on how to get the actual observations by another protocol [Di et al., 2007].

Using JPEG2000 and JPIP in WCS

WCS is mainly intended to serve fragments of raster files at different resolutions. JPEG2000 is an excellent format for saving the image on the server side, since JEPG2000 makes possible to generate a fragment of the image without decompressing the whole data, with some restrictions. It is also an excellent format for transmitting the image because it achieves a high compression ratio. In a WCS server, the JPEG200 transcoder determines which codeblocks are needed to extract any particular area, reads the codeblocks without uncompressing them and recombines them in a resulting file. Assuming that the user has also requested a JPEG2000 image, these codeblocks can be directly rearranged (transcoded) in a responded image. Unfortunately, this operation will only generate a resolution equal or submultiple to the original resolution of the image (in a range of resolution levels). For that reason, the client will receive a JPEG2000 image that normally exceeds the requested area and resolution, so in order to show the image on the screen, it has to decompress each codeblock and probably clip the image and resample it to the desired resolution. This will give a client an extra work that could be avoided if the requested resolution coincides with one of the resolution levels of the original image. This will hardly be the case if the client does not receive any information about the source exact resolution and how many transformation levels (resolutions) are there. Some OGC interoperability experiments have explored the possibility of communicating to the client the available resolutions. Figure 2 shows a possible combination of a WCS server with a JPEG transcoder and a JPEG2000 image; in this combination, the requested image is received by a WCS server that uses a transcoder to recover parts of the JPEG2000 image. Then, the WCS server can choose to serve the image as is (and possibly serve it in a resolution and bounding box close to the requested ones but not equal to them) or to decompress the image, clip it and interpolate it to the exact requested resolution and recompress it in a new JPEG2000 file before sending it to the client. The client could choose which process the WCS server will do using a vendor specific parameter.

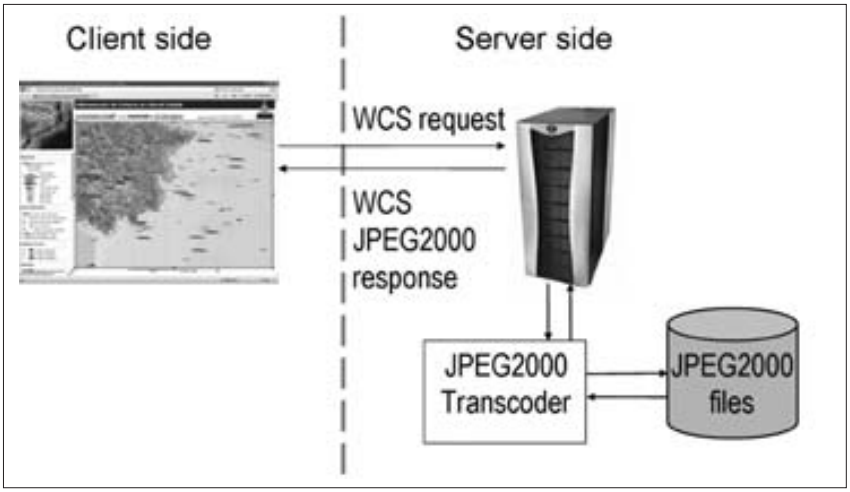


Figure 2 – JPEG2000 transcoder as a base for WCS services.

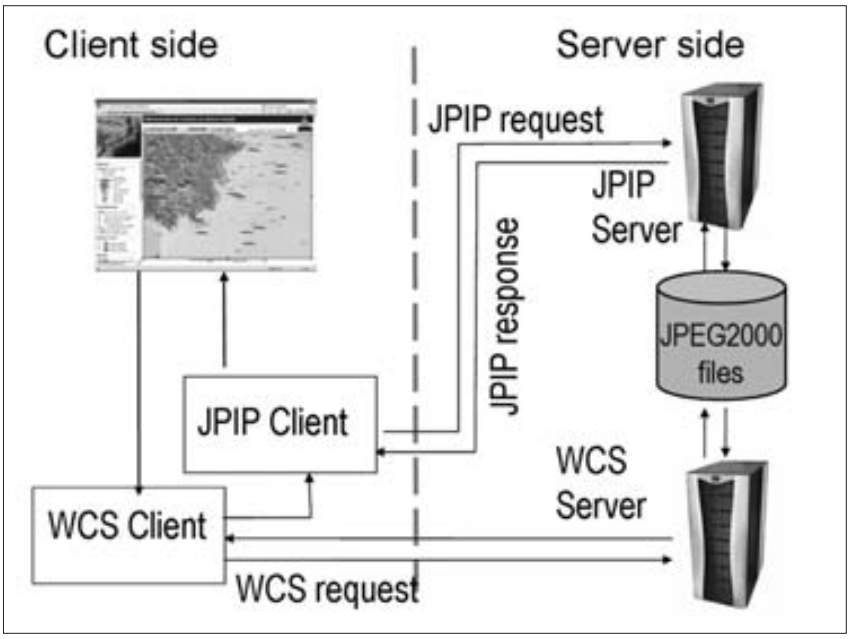


Figure 3 – WCS to JPIP transformation on the client side.

Figure 3 shows another possibility where a WCS client is combined with a JPIP client that communicates with a JPIP server. The advantage of this strategy is that we could use any compatible WCS client to dialog to a JPIP on the client side that maintains the codeblock cache and saves bandwidth. An extra piece of software is required to translate the WCS

requests (bounding boxes and resolutions) to the JPIP client parameters (columns and rows indices and offsets).

These combinations of servers do not require any restriction on the WCS protocol because a WCS response payload is reprocessed before reaching the WCS client. To eliminate the need of reprocessing and take full advantages of the JPEG2000 format, Giacovelli [2007] suggests some small modifications on the WCS. He discusses the possibility of asking directly for some resolution levels instead of using WCS resx/resy, or GridOffset parameters. This will avoid the need to interpolate to a requested output resolution in the server side and will provide a faster response. This is also less flexible and makes the client software more complicated.

In the OWS-4 OGC interoperability experiment, a WCS and a JPIP server were successfully combined [Gerlek, 2006]. The combination (see Fig. 4) requires 4 components: a JPIP client and JPIP server, and a WCS client and WCS server. WCS is used to discover the service and to translate the WCS request to a JPIP request. First, a WCS client requests a WCS layer specifying a bounding box and a resolution. Then, the server responds to the client with a JPIP requests that the client directly submits to the JPIP server, which responds to the JPIP client that shows the image.

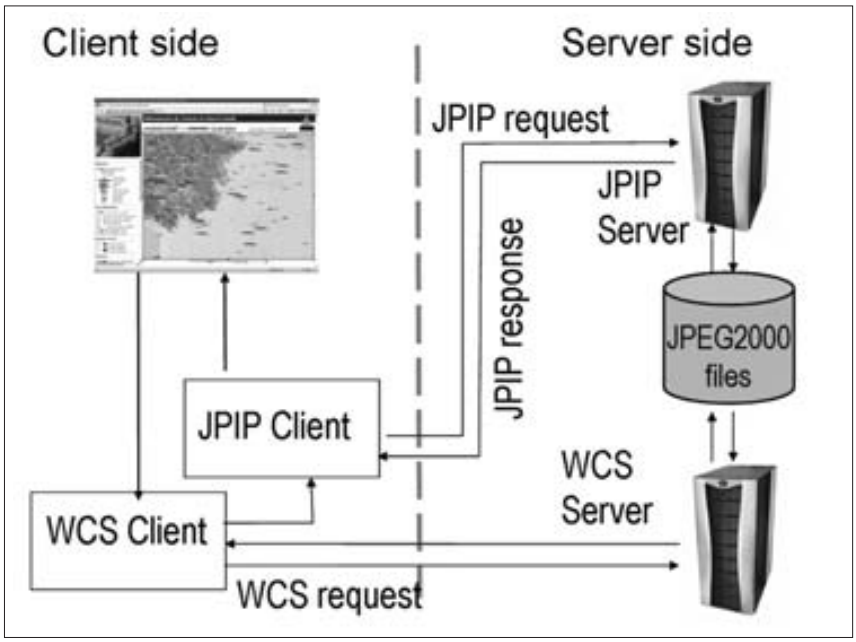


Figure 4 – WCS to JPIP transformation on the client side.

The OWS-5 OGC interoperability experiment defined a workflow where a sensor produces imagery that can be accessed through a JPIP server [Keens, 2008]. This JPIP server communicates a transactional WCS that a new layer exists and transmits it to the WCS server that registers this new layer in the service metadata document. Then the new layer can be immediately retrieved from the WCS service.

Using JPEG2000 in WMS and WMTS services

WMS is the most popular OGC standard. It defines how to communicate screen views of a map from the server to the client. A WMS GetMap request specifies the bounding box (in reference units) and the number of columns and rows of the map image. The returned map can be mostly encoded in a common format like PNG or JPEG but any other image format can be returned, like JPEG2000. GetMap operation allows requesting any image resolution so the server has to retrieve the relevant data from the original data and to render it to this particular resolution. WMS server uses source data, generally in a GIS format to render the map dynamically; this can be done to any render resolution from vector data but if the server has to deal with image datasets, it has to interpolate the image from the source resolution to the requested resolution. In the previous WCS examples, the response can still contain geo-reference information in a metadata file or any other form, so the client gets the needed extra information and react appropriately if the server does not respond exactly the resolution or extent that the client has requested. That is not the case for a WMS server that has to respond exactly what the client requires because the client has no way to check it. For that reason, a WMS server, that has internal data in a JPEG2000 format, has to decompress the image to the closest resolution and then clip and interpolate it to generate the exact resolution requested.

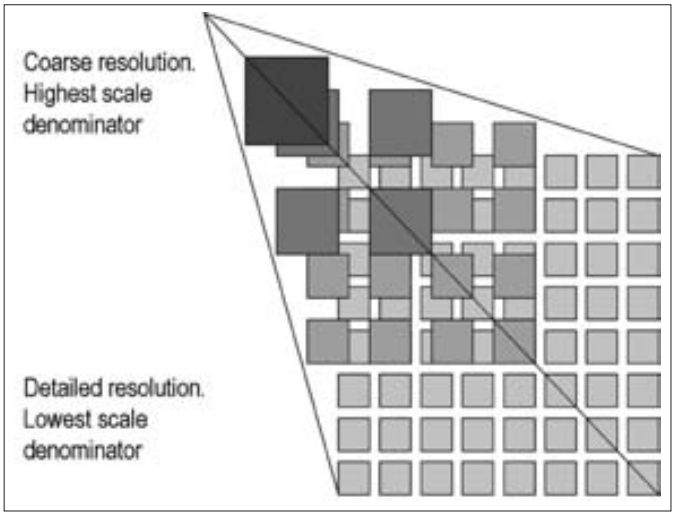


Figure 5 – Scale levels and tile matrices in WMTS.

Then, the server has to generate the requested image format. Generating a JPEG2000 image requires much more time than generating a JPEG image and the compression ratios that give satisfactory image fidelity are only a bit better in JPEG2000; so that, we only recommend JPEG2000 on the server side but not as an exchange format in WMS.

WMTS is a different scenario. In WMTS, the server has to specify a list of scales (called the tile matrix set) where maps can be served and a tile size and an origin of each scale (each tile matrix). The tile matrix set defines a pyramid of resolutions (see Fig. 5). For a WMTS based on internal raster data, the lower resolution is set to the original resolution of

the dataset and the other resolutions can be obtained by some interpolation technique like a Gaussian pyramid [Burt and Adelson, 1983]. However, in a WMTS server that uses internal JPEG2000 format the internal resolution levels are already created during the DWT and if they are used, this will avoid any need of resampling and interpolating data. In the service metadata document, a WMTS layer that is internally stored as a JPEG2000 will advertise the base resolution (the most detailed resolution) and powers of two of this basic resolution in a list of scales that will coincide with the transformation levels internally stored in the JPEG2000 file.
WMTS client requests only a regular piece of information from a particular scale level using the tile column and the tile row indices of the tile matrix defined for this scale (see Fig. 6).

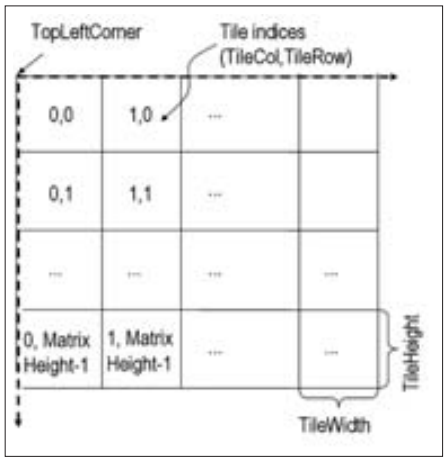


Figure 6 – TileMatrix in WMTS scale level.

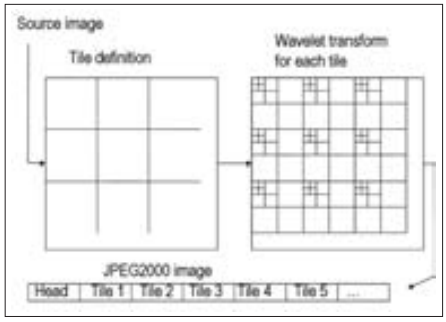


Figure 7 – Tiles in a JPEG2000.

The extraction of a small region of a JPEG2000 file is a complex process that can take time because the organization of the file is based on codeblock in the wavelet transformed space, so that decompressing a small portion of the original image can involve many codeblocks related with different transformation levels that are spread around the JPEG2000 codestream. Fortunately, JPEG2000 also has a similar concept that also permits to divide the image into tiles (see Fig. 7), so internally dividing the image in JPEG2000 tiles solves this problem.

In a tiled JPEG2000, each tile is compressed independently and is stored in an independent part of the JPEG2000 codestream [Gormish and Barnerjee, 2003]. Nevertheless, there are differences between the JPEG2000 tile model and the WMTS tile matrix set. In a JPEG2000 there is only one tile matrix define at the base resolution while WMTS tiles are defined at every resolution level. To make the extraction of a tile form a JPEG2000 easier, the image stored on a server site should have the same tile structure in the base resolution scale (lowest scale denominator) of the WMTS layer.
Since JPEG2000 defines only a tile structure for most detailed resolution, a JPEG2000 based WMTS server has to define the next scale (with scale denominator double of the previous one) with the same origin (TopLeftCorner) and the same tile width and height in pixels (that results in a tile that covers a region double of the previous scale level). Annex E of the WMTS standard already provides two well known scale sets that follow this rule.
Following this schema, the extraction of the base resolution level WMTS tile will involve one JPEG2000 tile that needs data from all internal JPEG2000 resolution levels. The extraction of a tile of the next WMTS scale level will require reading data from 4 JPEG2000 tiles but accessing codeblocks from all JPEG2000 resolution levels except the most detailed one. The extraction of a tile of the following scale level will require reading data from 16 JPEG2000 tiles but accessing data from all JPEG2000 codeblocks of all resolution levels except the 2 more detailed, and so on (see an example in Tab. 1).

Table 1 – Number of JPEG2000 tiles and transformation levels involved in the extraction of a single WMTS tile as a function of the output pixel size resolution (supposing a 5 transformation level JPEG2000).

Pixel size	N. of JPEG2000 tiles	Transformation levels involved
1m	1	1,2,3,4,5
2m	4	2,3,4,5
4m	16	3,4,5
8m	64	4,5
16m	256	5

In some cases, where many WMTS resolution levels are needed and huge images are involved, coarser resolution levels can be difficult to obtain from a complete JPEG2000 file because the number of JPEG2000 tiles involved. For that reason, we recommend to generate some (2, 3…) independent images of chained resolutions and combine them together on a single WMTS layer (see an example in Tab. 2).
A WMTS server could offer common formats like PNG or JPEG, but it can also provide native JPEG2000 format. In this case, the response image does not need to be uncompressed and recompressed to be served. A base resolution level WMTS tile can be obtained extracting the JPEG2000 tile codestream and generating a new header for this single tile. Other resolution levels will require extracting several JPEG2000 tile codestreams and transcoding them to a coarser resolution (eliminating the more detailed resolution levels).
In conclusion, a JPEG2000 file that will be used in a WMTS will perform much better if

Table 2 – Number of JPEG2000 tiles and transformation levels involved in the extraction of a single WMTS tile using 3 chained resolution images. Pixel sizes follow the WMTS GoogleMapsCompatible well know scale set

Pixel size	N. of JPEG2000 tiles	Transformation levels involved
JPEG2000 1.19*m pixel size (1.5Gb)*		
1.19m	1	0,1,2,3
2.39m	4	1,2,3
4.78m	16	2,3
9.55m	64	3
JPEG2000 19.11*m pixel size (5.9Mb)*		
19.11m	1	0,1,2,3
38.22m	4	1,2,3
76.44m	16	2,3
152.87m	64	3
JPEG2000 305.75*m pixel size (22.9Kb)*		
305.75m	1	0,1,2,3
611.50m	4	1,2,3
1222.99m	16	2,3
2445.98m	64	3

it is prepared (compressed) with the small tiles typically used in a WMTS service (*i.e.*, 256x265 or 512x512). When many resolution levels are needed a few number of resolution chained JPEG2000 images are recommended. Then, this structure is exposed in the layer description of the WMTS service metadata document as a base scale level and the next resolution levels will share the same origin and sizes up to the number of resolution levels

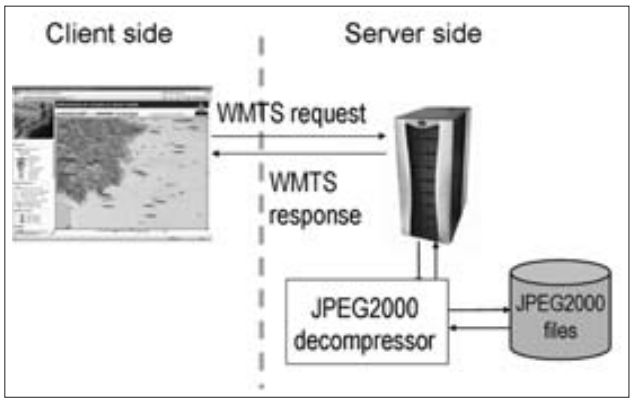


Figure 8 – WMTS based on JPEG2000 images on the server.

on the JPEG2000 files (the client is not aware of that diversity of file inputs). Then a WMTS server that receives a WMTS request will have to translate it to JPEG2000 decompression

routine commands (see Figure 8). If we choose the Kakadu decompressor (considered one of the fastest available) we will need to translate the WMTS parameters to Kakadu syntax (-reduce <discardLevels> and -region {<top>,<left>}, {<height>,<width>}).
The following expressions translate between WMTS standard request indices (tileRow,tileCol) to the Kakadu decompressor parameters:

$$
\begin{aligned}
top &= \frac{tileHeight}{imageHeight} 2^{scale\ Level}\, tileRow \\
left &= \frac{tileWidht}{imageWidth} 2^{scaleLevel}\, tileCol \\
height &= \frac{tileWitdh}{imageWitdh} 2^{scaleLevel} \qquad [1] \\
width &= \frac{tileWidth}{imageWidth} 2^{scaleLevel} \\
discardLevels &= scaleLevel
\end{aligned}
$$

where:
top left height and width = Parameters of the Kakadu decompressor syntax that are real numbers within the interval from 0 to 1 (top+heigh and left+width must not exceed 1). When truncated, top and left must be up-rounded and height and width down-rounded.
TileHeight and TileWidth = the size of the WMTS tiles in pixels imageHeight and imageWidth = Size of the complete decompressed image in pixels scaleLevel = A number that represents the scale level: 0 for the base scale level, 1 for the next scale level, 2 for the next scale level, etc.
tileRow and tileCol = WMTS tile indices.

Comparing WMTS and JPIP services

Generally speacking, JPIP protocol could be considered an alternative to WMTS since they share the common interest of disseminating images over the net in a fast way. There are 2 main differences between WMTS and JPIP: The first relays on the fact that WMTS typically uses common formats (PNG, JPEG) and can eventually use JPEG2000, but JPIP uses JPEG2000 only. For that reason, WMTS can be used in a web browser client without any extra software and this makes it ideal for combining it with other web applications, but JPIP requires a thicker and more specialized client. The second difference is that JPIP protocol allows maintaining codeblock cache on client and server, so that codeblocks received in previous transmissions do not have to be transmitted again. This property saves time and bandwidth when requesting a different zoom level from a region of the image that was previously requested with another resolution but requires the server knowing about the state of each client, either keeping a session on the server or by transmitting the client state to the server in each request. Current implementations of WMTS cannot do that and, in the same circumstances, they transfer some redundant information. Nevertheless, WMTS is designed as a completely stateless protocol, so that it can take advantage of the web caching mechanism that will eventually benefit all the WMTS concurrent users of a particular server, making WMTS more appropriate that JPIP for very popular web map services. Also, WMTS has a RESTful interface that allows deploying a WMTS service in a standard web server software (*i.e.* Apache, IIS, etc) as a set of static prerendered files,

with no extra software needed (making the deployment process compatible with all current Internet service providers).
Future implementations of the WMTS standard could be extended in a way that eliminates some redundant transmissions if a WMTS client has previously received a tile image for the same region of a coarser resolution without breaking compatibility with the current WMTS specification. A possible technique will consist in transmitting the coarsest resolution tile and then, when a more detailed resolution is requested, only the differential information needed to enrich the previous tile to this new level of detail. This can be achieved by using a Laplacian pyramid described by Burt *et al.* [1983]. Instead of storing true images for each level of the pyramid (like the Gausian pyramid does), in a Laplacian pyramid, the information stored for each particular level is constructed as the difference between the Gausian image of this level and the Gausian image of the next coarse resolution. Having a coarser image and the Laplacian information for the next level it is possible to reconstruct the actual image for the next level. However, a WMTS service that provides Laplacian resolution levels will require a thicker and specialized client that stores previous coarser resolutions and combine information from the next levels to show the actual image. This approach will have a set of requirements similar to a JPIP client and will not have the extra JPIP protocol advantages in quality scalability, resolution scalability, and flexible spatial random access [Patel *et al.*, 2007] but it will still remain a stateless solution that will properly use internet caching mechanisms and have better concurrency scalability.

GMLJP2

GML in JPEG 2000 for Geographic Imagery (GMLJP2) specifies how to encapsulate metadata and annotations in a JPEG2000 file [Kyle et al., 2006]. As any image file, a JPEG2000 can be used to contain geospatial information in a raster form but geospatial information requires a minimum set of extra metadata that cannot be directly encoded in a JPEG2000 header, like the bounding box in a coordinate reference system or the coordinate reference system itself. Fortunately, JPEG2000 has a mechanism called XML boxes that is used in the GMLJP2 standard to include Geography Markup Language (GML) features, metadata (*e.g.*, coordinate reference systems and units of measure), annotations, and styles [Kyle et al., 2006]. Any geospatial JPEG2000 should use this extension of the JPEG2000 standard to include at least the bounding box, the coordinate reference system and a reference to an ISO19139 metadata document. This recommendation particularly affects the JPEG2000 products that use WCS and SOS standards (considered before). A useful characteristic of GMLJP2 is the ability to embed vector annotations in a JPEG2000 file, that could be used to add to the image some additional features, like training areas, landmarks or to include textual descriptions. A simple JPEG2000 viewer will not show them but will be still able to show the image.

Conclusions

JPEG2000 can be combined with several OGC web services. We have explained some strategies that can be used to merge JPEG2000 or JPIP with WCS or SOS. We have focused on the new OGC WMTS protocol and in implementations that use JPEG2000 images as an internal storage format. Even if JPEG2000 images have a codeblock structure to easily extract a region of the image at any resolution, in order to have the fastest response possible,

JPEG2000 needs to be formatted as a tiled JPEG2000. WMTS scale sets and tile matrices can be restricted using a small set of rules that will allow the server to faster respond in common formats or in even in small JPEG2000. Moreover, a corporation can use a data storage with a collection of JPEG2000 files that can be used as internal repository for WMTS, WCS or SOS data services and, at the same time, directly used in the corporative RS or GIS application completely avoiding duplicity of data formats and providing a highly efficient compressed storage format (see Figure 9).

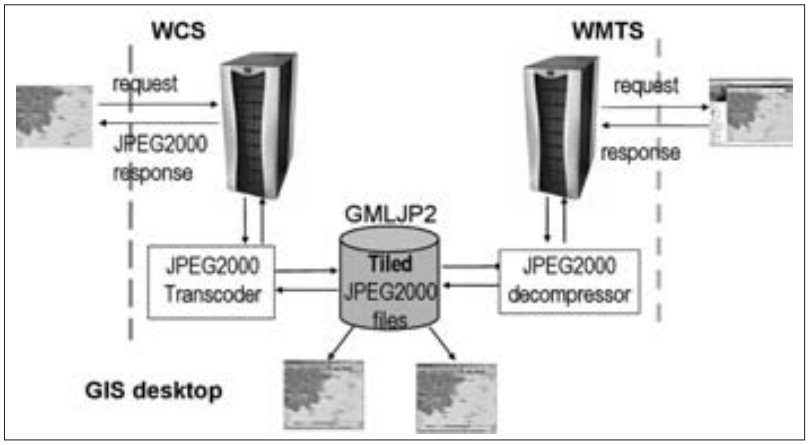


Figure 9 – Tiled JPEG2000 images repository is the only storage needed for use either in GIS desktop, in coverage servers or in tile map servers.(triple line takes advantage of the Internet caching mechanisms).

Acknowledgements

This work has been supported by the Spanish Government, by FEDER, and by the Catalan Government, under Grants TSI2006-14005-C02-02, TIN2009-14426-C02-02 and GRUMETS 2009 SGR 1511, and also by the European Commission through the FP7-242390-GEO-PICTURES project (FP7-SPACE-2009-1)

References

De la Beaujardiere, J. (2004) – *OGC Web Map Service (WMS) Interface*. Ver.1.3.0, OGC 03-109r1.

Burt P.J., Adelson E.H. (1983) – *The Laplacian Pyramid as a Compact Image Code.* IEEE Transactions on Communications, Vol. Com-31, No. 4.

Christopoulos C., Skodras A., Ebrahimi T. (2002) – *The JPEG2000 still image coding system: an overview*. IEEE Transactions on Consumer Electronics, Vol. 46, No. 4, pp. 1103-1127.

Di, L., Yu G., Chen N., Min M., Wang H. (2008) – *Support of Asynchrony in Sensor Web.* Earth Science Technology Conference 2008. University of Maryland.

Gerlek M.P. (2006) – *OWS-4 IPR for WCS Support for JPEG 2000*. OGC 06-128.

Giacovelli P., Gerlek M. P. (2007) – *WCS 1.1 Application Profile for JPEG 2000 Coverage Encoding*, version 1.0, OGC 07-145.

Gormish M., Banerjee S. (2003) – *Tile-Based Transport of JPEG 2000 Images*. N. Garcıa,

J.M. Martınez, L. Salgado (Eds.): VLBV 2003, LNCS 2849, pp. 217–224.
ISO/IEC 15444-1 (2000) – *JPEG2000 image coding system* – Part 1: Core coding system.
ISO/IEC 15444-9 (2003) – *JPEG 2000 image coding system* – Part 9: Interactivity tools, APIs and protocols.
Keens S. (2008) – *OWS-5 WCS JPIP Sub-setting Engineering Report*. OGC 07-169.
Kyle M., Burggraf D., Forde S., Lake R. (2006) – *GML in JPEG 2000 for Geographic Imagery (GMLJP2)*. Encoding Specification. OGC 05-047r3.
Masó J., Pomakis K., Julià N. (2010) – *OGC Web Map Tile Service (WMTS)*. Implementation Standard. Ver 1.0, OGC 07-057r7.
Na A., Priest M. (2007) – *OGC Sensor Observation Service SOS*. Ver. 1.0.0 OGC 06-009r6.
Patel S., Yuval S., Lau D. (2007) – *Interactive Image Browsing with JPEG2000/JPIP*. Master project in Image Communication I. Stanford University. , URL: http://www.stanford.edu/class/ee398a/projects/reports/LauPatelYuval.pdf (accessed 26 Nov. 2009).
Rajabifard, A., Mansourian, A., Valadan Zoej, M.J., Williamson, I.P. (2004) – *Developing Spatial Data Infrastructure to Facilitate Disaster Management*. Proceedings of GEOMATICS'83 Conference, 9-12 May, Tehran, Iran.
Taubman D.S., Marcellin M.W. (2002) – *JPEG2000: Image compression fundamentals, standards and practice*. Kluwer, Academic Publishers.
Whiteside A., Evans J.D. (2008) – *OGC Web Coverage Service (WCS)*. Implementation Standard. Ver. 1.1.2, OGC 07-067r5.

Received 06/03/2010, accepted 28/09/2010

6. IMPACT OF USER CONCURRENCY IN COMMONLY USED OPEN GEOSPATIAL CONSORTIUM MAP SERVER IMPLEMENTATIONS

Aquest capítol és una reproducció de: Masó J., Diaz P., Pons X., Monteagudo-Pereira J. L., Serra-Sagristà J. and Aulí-Llinàs F. (2011), Impact of User Concurrency in Commonly Used Open Geospatial Consortium Map Server Implementations. *The International Conference on Advanced Communications and Computation* (INFOCOMP 2011). Rückemann CP, Christmann W, Pankowska M. (Eds.), IARIA, Barcelona. 23 al 29 d'octubre. ISBN: 978-1-61208-161-8 (en aquest document es referencia com Masó *et al.*, 2011a)

Impact of User Concurrency in Commonly Used Open Geospatial Consortium Map Server Implementations

Joan Masó, Paula Díaz, Xavier Pons, José L. Monteagudo-Pereira, Joan Serra-Sagristà, Francesc Aulí-Llinàs
Universitat Autònoma de Barcelona, Spain
joan.maso@uab.cat, paula.diaz@uab.cat, xavier.pons@uab.cat, jlino@deic.uab.cat, joan.serra@uab.cat, fauli@deic.uab.cat

***Abstract*—In emergency situations, it is of paramount importance that accurate assessments showing what is happening in the field, and when and where it is happening, are distributed quickly. This increases the aid workers' awareness of the situation and can be used to organize the workers more efficiently. The increasing number of satellite images available means that new data can be obtained rapidly and the information can be kept constantly up to date. These data can be distributed easily using open standards over the Internet. In a large post-disaster event, the demand for information increases dramatically, which can negatively impact the performance of the services provided. Here, we assess seven of the most popular server solutions (GeoServer, MapServer, MiraMon Map Server, Express Server, ArcGIS Server, TileCache and GeoWebCache) for map service standards (WMS, WMTS, WMTS-C, TMS), and compare their response times, user functionalities and usability.**

Keywords-server; WMS; tile; response time; cluster; performance.

I. INTRODUCTION

Nowadays, there is an increasingly large amount of data, software and geographic standards available (public, private and voluntary), which allow satellite data to be used in a wider range of consolidated and specialized areas and applications. Current space technologies, such as meteorological and earth observation satellites integrated in global networks like GMES, communication satellites and Global Navigation Satellite Systems (GNSS) combined with Geographical Information Systems (GIS) [1], hazard modeling and analysis have also contributed to this increase in applications and data. Nevertheless, better spatial, temporal (synergies, constellations, etc.) and spectral resolutions of remote sensing imagery generate a huge amount of data that is difficult to store, discover, analyze and distribute. Heaps of tapes, CDs and hard drives full of data have been replaced by web-based data dissemination infrastructures that make searching and discovery easier. Web portals and clearinghouses increasingly implement standardized protocols and are integrated into a larger System of Systems, like GEOSS [19].

Despite the number of map server implementations that claim to be the fastest and the most robust on the market, there are few studies that apply rigorous metrics to determine the real performance of the servers or compare strategies to increase their performance.

This paper is an extension of a previous article [11] that evaluates the efficiency and possibilities of several map servers (i.e., MapServer [7], GeoServer [4], MiraMon Map Server [18], Express Server [6], ArcGIS Server [3], TileCache [13, 8] and GeoWebCache [15]) that implement international standards (e.g., Web Map Server (WMS [2]), Web Map Server – Cache (WMS-C [13]), Web Map Tile Server (WMTS [9])) connected to satellite image repositories. This research was carried out in the context of GEO-PICTURES, an acronym for *GMES and Earth Observation combined with Position based Image and sensor Communications, Technology for Universal Rescue, Emergency and Surveillance*. GEO-PICTURES is an EU FP7 SPACE project that aims to integrate satellite imagery into in-situ sensors and geo-tagged media (photos and video) to create a tool for decision making in emergency disaster situations. The complete GEO-PICTURES solution covers the capture, transmission, and analysis of data, which is re-elaborated and re-distributed to the aid forces as well as to the general public using several web platforms.

This article begins with the description of the materials and methodology used to perform this study, followed by a through explanation of the tests applied to the performance of the servers. Here an evaluation of concurrent requests to a single server and to a cluster of servers is done. The article continues with the comparison of different standards, this is what it is called: tiling the request and response. Finally, it concludes with a section where the most relevant results are discussed.

II. MATERIALS AND METHODOLOGY

Trials were performed with 22 GeoEye-1 (Orthorectified GeoTIFF; provided by Google [5]) imagery datasets that form a 4Gb raster of 40994 x 57392 pixels, covering Port-au-Prince and surroundings, on 16 January, 2010, 3 days after the earthquake. The influence of scale, intensive client use, and image size and format (JPG, GIF and PNG) were studied for the WMS, WMS-C and TMS [16, 17] protocols. As possible solutions to concurrent requests, we evaluated the efficiency of the Internet cache as well as a cluster of servers in an NLB (Network Load Balance) configuration. The seven servers were set with the minimum configuration required to be run, i.e., without any extra preparation of the data. In order to shield probabilistic error caused by network latency and other uncertain factors, the products were considered to be deployed and requested by local clients [14].

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-161-8

In order to guarantee the comparability of the results, the seven server software (Table 1) were installed on the same computer, which acted as a common server (Intel® Core™ i3 CPU 540 @ 3.07 GHz, 3.06 GHz, 2.92 GB de RAM. Microsoft Windows XP Professional, version 2002 Service Pack 3). Requests were randomly generated from clients in the local network.

TABLE I. LIST OF THE SERVERS EVALUATED AND THEIR SPECIFICATIONS.

Server	Specifications	Date
MapServer	version 5.6.3 over Apache web server version 2.2.15	April 20, 2010
GeoServer	version 1.7.2 over Jetty web server version 6.1.8	January 19, 2008
ArcGIS Server	version 9.3.1 over Internet Information Service version 5.1	January 1, 2009
Express Server	version 6.1 over Internet Information Service version 5.1	July 1, 2008
MiraMon Map Server	version v. 7.0e over Internet Information Service version 5.1	July 26, 2010
TileCache	version 2.11 over Internet Information Service version 5.1	December 22, 2007
GeoWebCache	version 2.0.1 over Jetty web, 'build over GeoServer	January 20, 2010

MapServer, GeoServer and ArcGIS Server can work over the original GeoTIFF images without having to create an image mosaic. MapServer and GeoServer require a shape file to be created with rectangles representing bounding boxes of each raster file (index file), and ArcGIS Server requires Image Definition (ISDef), which makes it possible to use the original GeoTIFF image format provided by GeoEye-1. Express Server requires the image index to be in JPEG2000 or in the MsSID compressed format. MiraMon requires a full automated pre-rendering of the images in several resolutions to set up the images. This takes several minutes to complete and needs up to 33% extra disk space. TileCache and GeoWebCache can create pre-rendered tiles and save them in a cache for further use or generate them on the fly. Automatic on the fly generation is an advantage that can save time when a new layer is set up in an emergency situation.

All studied protocols request maps by creating an URL using specific standardized syntax. This URL is requested from the server and an image is obtained in screen resolution (in the case of error the server sends an exception message). The URL requests were randomly generated by a program and requested using a command line tool (an application called *Wininet*) that waits for the response, saves it and reports the total time spent on this particular communication. The response time was redirected to an archive that was converted into a table of values that were used to create the performance graphics that are shown here. This paper describes the methodology employed and the numerical and graphical performance metrics, and evaluates strategies for improving performance.

The randomly generated URL methodology employed guarantees that speed measures are comparable and independent from the selected bounding box or request sequence. Nevertheless, users in front of a computer screen browsing the maps do not generate random requests but rather they request regions next to the previous ones at the same zoom level (pan) or they zoom in and out in the same region. However, human browsing patterns are out of the scope of this work and will be considered in the future.

III. EVALUATION OF CONCURRENT REQUESTS TO A SINGLE SERVER

One of the main factors that affect the performance of web servers is the concurrency of requests. We measured both the influence of the pixel size and the image size on the response time for WMS requests. More than one hundred different requests were made from up to 6 simultaneous clients. The graph (Figure 1) shows the response time for different pixel size requests.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-161-8

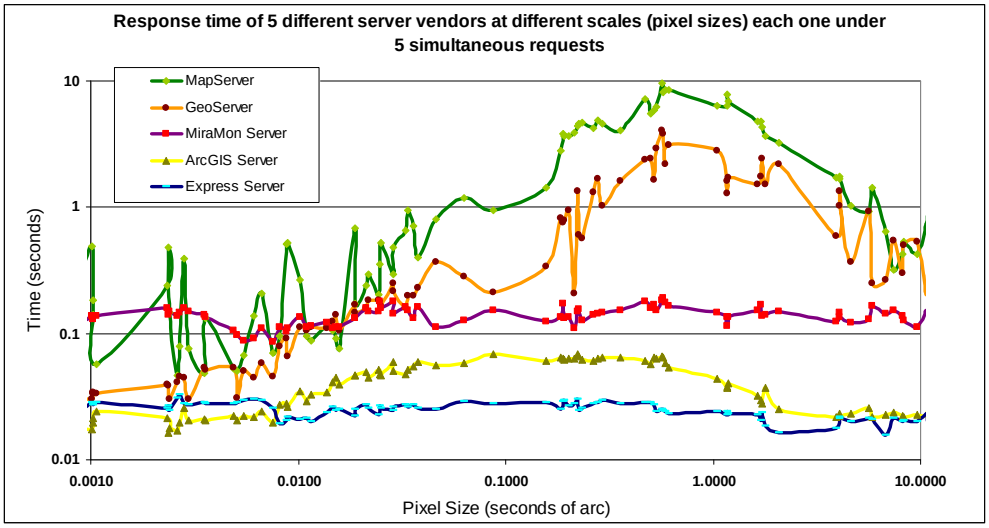


Figure 1. The response times of MapServer, GeoServer, MiraMon Map Server, ArcGIS Server and Express Server in relation to pixel size in concurrent requests from 5 simultaneous clients respectively.

One common aspect of the servers analyzed is that the response time increases as the number of simultaneous requests made by clients to a single server increases. The fastest server is Express Server, probably due to the nature of the wavelet compressed format (MrSID or JPEG2000) that is internally organized in a pyramid of zoom levels of the images used as a database. ArcGIS Server obtains the best results using original GeoTIFF images. MiraMon Map Server obtains intermediate results as it requires a pre-rendering process to generate tiles. MapServer is programmed in C language and GeoServer uses java code. MapServer performs faster when a single client is used, but GeoServer is faster than MapServer for concurrent requests. This could be because java provides better and easier multithread support. There are also small differences in response times depending on the output format requested. JPEG is the fastest format in MapServer, Express Server, ArcGIS Server and MiraMon Map Sever, but PNG is fastest in GeoServer.

A request with a pixel size that generates a map covering a region equivalent to the boundary of the GeoTIFF set (nearly 0.893 seconds of arc in range, width 443) obtains the slowest response time. A map with a smaller pixel size only shows a part of the GeoTIFF set area and obtains a faster response. A map with a larger pixel size leaves some blank areas, and also results in a faster response time.

IV. EVALUATION OF A CLUSTER OF SERVERS

In the current state of maturity of the hardware it is not possible to dramatically increase performance by getting a faster machine, even if you are willing to pay more. Current computer technology has reached a speed limit in CPU processing time and disk speed access. To overcome the performance degradation observed in concurrent requests a possible solution is to set up a cluster of servers that can act as a virtual single server that deals with requests in parallel. We carried out tests comparing a single WMS server (Figure 2) and a WMS computer cluster server (Figure 3), in which 6 computers are able to respond to different clients at the same time as if they were a faster single server. These tests (consisting in the same requests) were carried out with up to 17 simultaneous clients to evaluate the response time of the MiraMon Map Server.

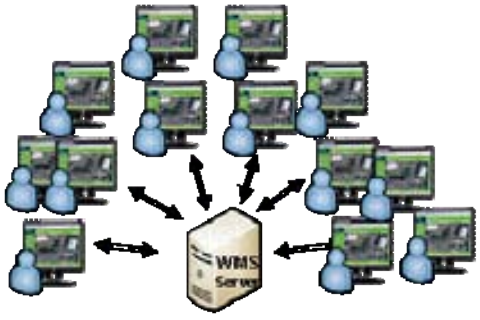


Figure 2. Computer single server structure.

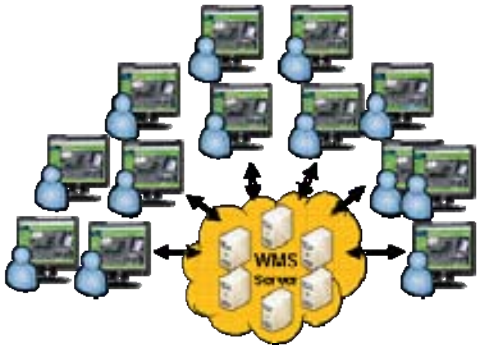


Figure 3. Computer cluster server structure.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-161-8

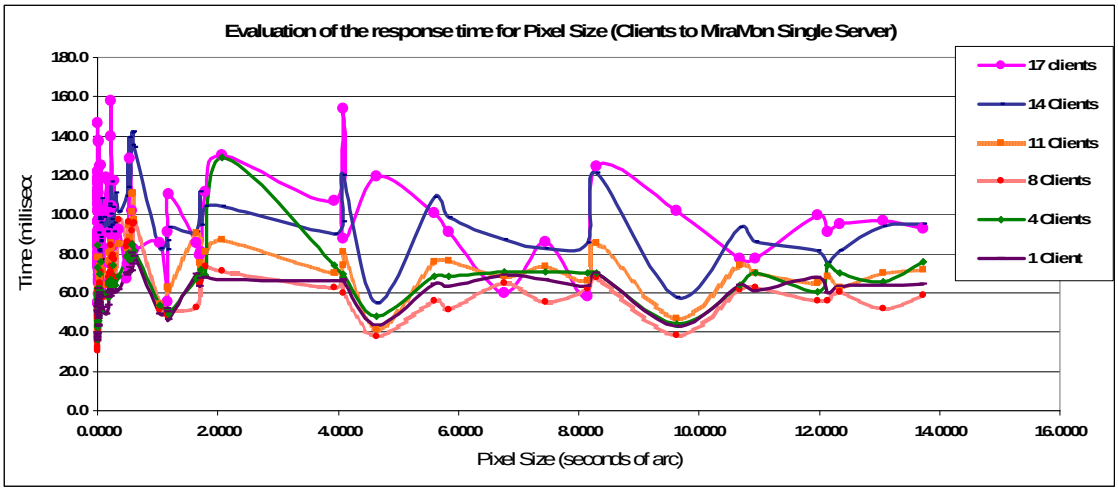


Figure 4. Response time for different concurrent requests for up to 17 clients of a single MiraMon Map Server.

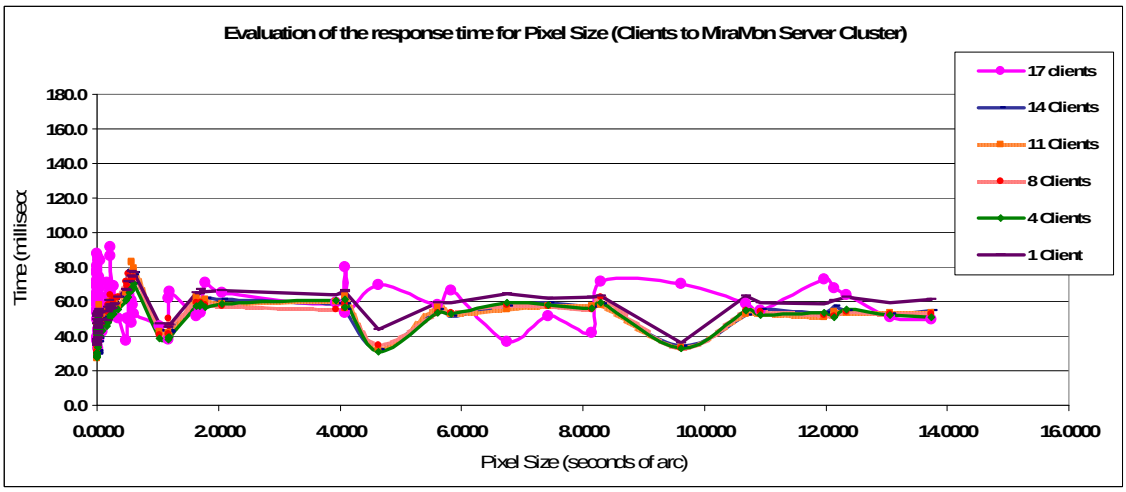


Figure 5. Response time for different concurrent requests for up to 17 clients of a cluster MiraMon Map Server.

Response time measurements comparing a single server (Figure 4) and a six-computer cluster (Figure 5) stressed with multiple client requests show that the response time of the NLB server is much more stable and almost equivalent to the single client stress case, even for 17 simultaneous requests. We expect some degradation if we increase the number of requests further, but fortunately the performance of the NLB cluster can be improved again by aggregating more servers to the cluster (up to 64 in Windows 2003). If we suppose that the performance is linear, this means that this configuration can be scaled to serve at least ~200 simultaneous requests without performance degradation. Note that the response time for these requests is always lower that 0.1 second, so this configuration is equivalent to 2000 requests per second.

V. TILING THE REQUEST AND RESPONSE

In the previous sections, we assumed a common WMS interaction in which a WMS client requests the entire image needed to cover the client viewport in a single piece. Some WMS clients (like OpenLayers) are now able to tile the space in a regular matrix of small pieces [12]. Therefore, several tiles are needed to cover the whole viewport but the client can recycle some tiles when the user moves the view laterally and can also take advantage of the cache mechanisms. However, this strategy can have its drawbacks if the caching mechanism cannot help and the server has not been prepared to manage this situation because, as we have discussed previously, the response time can increase even if each tile is smaller than the whole view. However, users do not perceive this because some tiles get to the client sooner and are shown immediately. This paper clarifies the effects of this approach on a classical WMS server and quantifies the difference between fast full image delivery (WMS) and tiled image delivery (WMS-C and TMS). We also studied improving these situations by applying tile strategies directly to the server, like OSGeo WMS-C and OGC WMTS (10).

We carried out speed metrics in 3 different services for 7 servers: Express Server, ArcGIS Server, MiraMon Map Server, GeoServer, MapServer, TileCache

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-161-8

combined with MapServer and GeoWebCache combined with GeoServer. We simulated tiled clients (tiles of 256x256 pixels) that make requests to common WMS (which have no particular strategy for dealing with tiles) in the three following configurations:

- Tiled WMS in unlimited concurrent requests. This consists in requesting all the tiles needed to cover the viewport at the same time. In our case, from 6 to 12 tiles were needed to cover the entire viewport requested. In some cases, this resulted in momentary server saturation (Figure 6), like in MapServer and GeoServer. The three servers with the best performance were Express Server, ArcGIS Server and GeoWebCache.

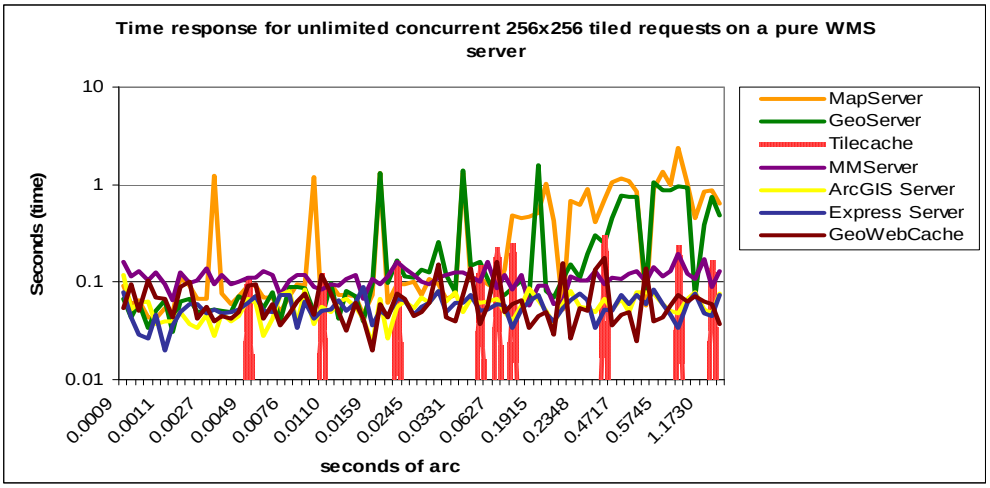


Figure 6 . WMS-C for unlimited concurrent tile requests.

- Tiled WMS in semi-concurrent requests. This consists in limiting simultaneous requests to the maximum number of requests to a server that a web browser allows (e.g., Firefox 3.6 allows 6 simultaneous petitions but Internet Explorer 6.0 only allows 2). In our case, we used a mean value of four tiles at the same time, then we waited until the server finished to request the next four (Figure 7). Some servers performed better compared to the previous case, such as MapServer, GeoServer and MiraMon, while others performed worse. Tiled servers performed better in general.

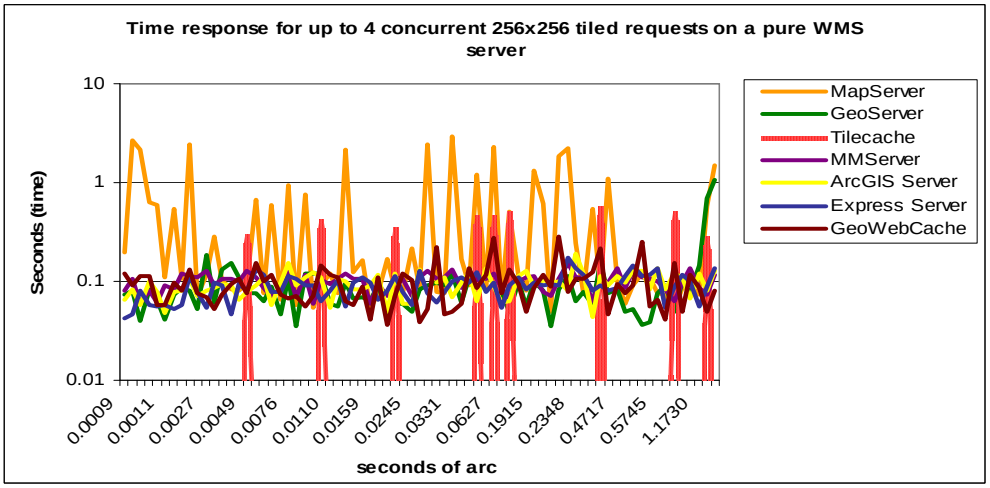


Figure 7.WMS-C for semi-concurrent requests, up to 4 concurrent tile requests.

- Tiled WMS in sequential requests. This consists in requesting each tile after the previous request has been completed (Figure 8). This results in a more stable response time but it is not the optimum situation, especially for GeoServer, MiraMon Server and in some cases ArcGIS Server. GeoWebCache has the best performance.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-161-8

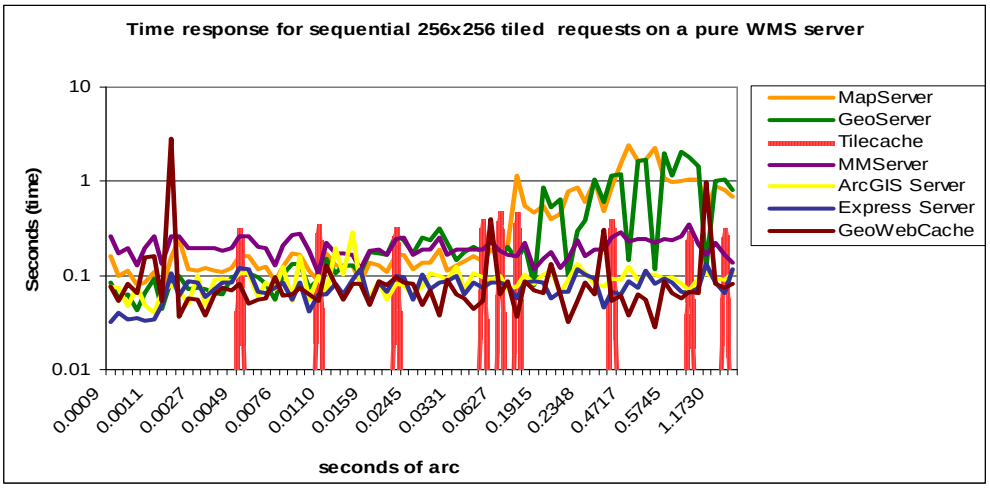


Figure 8. WMS-Cached for sequential requests.

Finally, we compared these three configurations with a regular WMS full viewport image (Figure 9), and evaluated performance degradation. TileCache and GeoWebCache are not represented in Figure 9 because these servers are not able to respond to a full WMS viewport.

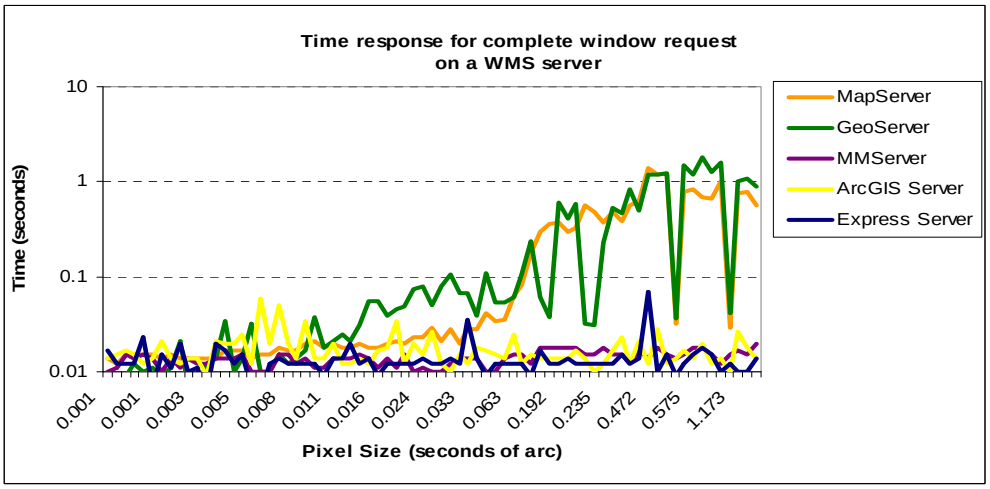


Figure 9. Regular WMS for full image requests.

Figure 9 shows that a full WMS viewport is the fastest for all servers, particularly for Express Server, ArcGIS Server and MiraMon Map Server, probably because the server only does the work once even if the volume of information to deliver is bigger. When tiles are used, requesting all tiles sequentially results in the slowest solution for all servers; however, limiting the number of concurrent requests to 4 improves the response time significantly. This is the best performance situation for MiraMon Map Server and GeoServer. After seeing this, it is easy to understand why many web browsers limit the number of simultaneous requests to a relatively small number (depending on the product and the version). Out of the concurrent tile request situations, this is the best tile solution for MapServer, Express Server and ArcGIS Server. Determining the optimum semi-concurrence number for each server will be the focus of a future work.

The tile products tested (TileCache and GeoWebCache) provide a way of pre-rendering tiles or saving tiles that are generated on the fly for further use. The main drawback is the generation time, but this can be partially overcome by *metatiling* strategies. Both TileCache and GeoWebCache support metatiling. Instead of generating each tile individually, a *metatile* is generated, creating a single large map image that can be divided into a number of tiles. Figure 10 shows that for all servers, analyzing a 512 x 512 image requires more time, but much less time than that required for analyzing four 256 x 256 images. This is because generating a metatile involves accessing source data only once instead of four times for a set of four tiles

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-161-8

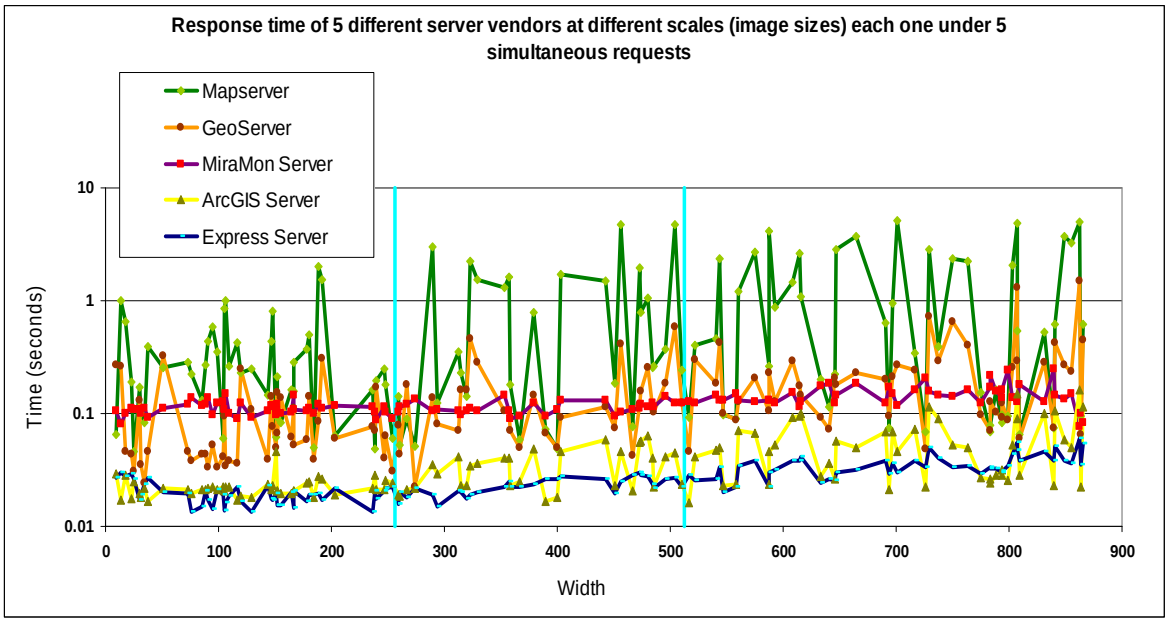


Figure 10. Response time for image size in concurrent requests to 5 single servers. Two marks at 256 and 512 pixel sizes have been added to facilitate the comparison between the response time in these two common pixel sizes.

VI. CONCLUSIONS

The speed tests described in this paper are a practical demonstration of the suitability of certain servers and service configurations in certain domains in which the reliability of the services under high stress conditions is imperative. This document summarizes and quantifies the results of our speed tests and determines which servers are faster under the minimum configuration.

All the analyzed servers have slower performances when the number of simultaneous clients is increased. A cluster server can be used to solve this situation: a group of computers is able to respond at the same time to different clients, assigning each client to a different computer in the group.

The results show that WMS servers do not perform well if clients using tile strategies are used over servers that are not optimized for tile response. MapServer and GeoServer with minimum data configuration do not require a data preparation process; however, they do not perform as well as other services that require indexing methods like MiraMon Map Server or Express Server. MapServer (based on C++ code) performs better than GeoServer (based on Java code) under single client requests, but GeoServer is surprisingly faster under concurrent simultaneous requests. The fastest WMS server is Express Server which works with MsSID or JPEG2000 compressed images that are 5% of the original size. The fastest tile server in the three cases assessed (concurrent, semi-concurrent and sequential requests) is GeoWebCache built over GeoServer.

ACKNOWLEDGEMENTS

This work was supported by the Spanish Government, by FEDER, and by the Catalan Government, under Grants TSI2006-14005-C02-02, TIN2009-14426-C02-02 and GRUMETS 2009 SGR 1511, and also by the European Commission through the FP7-242390-GEO-PICTURES project (FP7-SPACE-2009-1). Xavier Pons is the recipient of an *ICREA Acadèmia* Excellence in Research grant.

REFERENCES

[1] Altan, O., Backhaus R., Boccardo, P., and Zlatanova, S. (2010), Geoinformation for Disaster and Risk Management. Examples and Best Practices, Copenhagen, Denmark. ISBN 978-87-90907-88-4.

[2] De la Beaujardiere, J. (2004) – OGC Web Map Service (WMS) Interface. Ver.1.3.0, OGC 03-109r1. Web: http://www.opengeospatial.org/standards/wms, last access March 1st 2011.

[3] ESRI (2010), Image Server & ArcGIS Server, 9.3. 380 New York St., Redlands, CA 92373-81000 USA.

[4] GeoServer (2010) user manual version 1.7, Web: http://docs.geoserver.org/, last access March 1st 2011.

[5] Google Crisis Response (2010), Haiti Earthquake – GeoEye Satellite Imagery Download, Web: http://www.google.com/relief/haitiearthquake/geoeye.html, last access March 1st 2011.

[6] LizardTech (2010), LizardTech Inc y Lizardtech España SL. GeoExpress compressor. version 8 & Express Server™ version 6.1.

[7] Map Server Team, (2010), MapServer Release 5.6.5 documentation, Open source Web Mapping, Web: http://mapserver.org/tutorial/index.html, last access May 24th 2011.

[8] Marin, L., (2008) Setting up TileCache on IIS, in a blog at WorldPress.com Web: http://viswaug.wordpress.com/2008/02/03/setting-up-tilecache-on-iis/, last access March 1st 2011.

[9] Masó J., Pomakis K., and Julià N. (2010a) – OGC Web Map Tile Service (WMTS). Implementation Standard. Ver 1.0, OGC 07-057r7 Web: http://www.opengeospatial.org/standards/wmts, last access March 1st 2011.

[10] Masó J., Zabala A., and Pons X. (2010b) Combining JPEG2000 Compressed Formats and OGC Standards for Fast and Easy

Dissemination of Large Satellite Data, Italian Journal of Remote Sensing - 2010, 42 (3): 101-114

[11] Masó J, Díaz, P., and Pons, X. (2011) Performance of standardized web map servers for remote sensing imagery. In: Proceedings of "Data Flow: From Space to Earth. Applications and interoperability Conference" (21-23th March 2011, Venice).Corila - Consorzio per la Gestione del Centro di Coordinamento delle Attività di Ricerca Inerenti il Sistema Lagunare di Venezia. pp.83-83 ISBN:9788889405154.

[12] Matt Mills, NASA World Wind Tile Structure. Web: http://www.ceteranet.com/nww-tile-struct.pdf, last access March 1st 2011.

[13] MetaCarta Labs, TileCache Contributors (2006-2010), TileCache Web: http://tilecache.org/, last access May 24th 2011.

[14] Nie Y., Xu, H., and Liu H., (2011), The Design and Implementation of Tile Map Service, in Advanced Materials Research, Vol. 159 (2011) pp 714-719, Trans Tech Publications, Switzerland.

[15] OpenGeo (2011), GeoWebCache User Manual, Web: http://geowebcache.org/docs/current/, last access May 24th 2011.

[16] OSGeo (2011), Tile Map Service Specification, OSGeo Wiki. Web: http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification, last access March 1st 2011.

[17] OSGeo, OpenLayers wiki, Using Custom Tile Sources, Google-like Tile Layer Support, Web: http://trac.osgeo.org/openlayers/wiki/UsingCustomTiles, last access March 1st 2011.

[18] Pons, X., (2004) MiraMon. Geographic Information Systems and Remote Sensing software. v. 4, Center for Ecological Research and Forestry Applications, CREAF, Bellaterra. 286 p. ISBN: 84-931323-4-9.

[19] GEOSS, The Global Earth Observation System of Systems. Web: http://www.earthobservations.org/

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-161-8

7. RESUM DE RESULTATS / SUMMARY OF RESULTS

7.A. Resum de resultats (versió en català)

El resum de resultats d'aquesta tesi es realitza tot fent un repàs de les principals fites assolides obtinguts en les diverses publicacions que formen els capítols 2 a 6. En primer lloc es presenta els aspectes generals derivats de la implantació d'estàndards geospacials en les IDE i al SIG en general per, posteriorment, concretar en els aspectes específics de la navegació de mapes.

7.A.1. Aspectes generals derivats de la implementació d'estàndards geospacials

La recerca realitzada sobre el grau d'interoperabilitat de les infraestructures de dades espacials, els sistemes de sistemes i les tecnologies per a la terra digital (*digital Earth*) ha conduit a un conjunt de resultats que es resumeixen en aquesta secció i que han estat majoritàriament extrets i tractats en els capítols 2 i 3 d'aquesta tesi.

7.A.1.1. Aspectes conceptuals i d'arquitectura

En aquesta secció es resumeix els resultats més conceptuals i d'arquitectura que discuteixen les febleses trobades tant en els estàndards com en les seves implementacions; alhora, s'explora les mesures proposades per a millorar aquesta situació i la interoperabilitat geospacial en general.

7.A.1.1.1. Resultats de l'anàlisi de les Infraestructures de dades espacials.

Els resultats presentats en aquest tema són un conjunt de recomanacions per a millorar les implementacions de les IDE com per exemple, afegir més funcionalitats, millorar la interoperabilitat o millorar el rendiment. Aquestes recomanacions han estat classificades en els següents punts: les metadades sobre les dades i serveis, models de dades, descàrrega de dades, dades i serveis web de processament, la representació i la simbolització, i el mercat de masses. Els estàndards considerats en aquest estudi es representen a la figura 11.

En proporcionar aquestes recomanacions es considera el costat de l'usuari i el costat del servidor, de forma paral·lela. Aquest resum recull i comenta algunes d'aquestes recomanacions, però la taula 2 en el capítol 2 en conté un conjunt més complet.

Millora de les metadades sobre les dades

Els estàndards de catàlegs de metadades es presenten en una varietat de versions i perfils, tant del CSW ISO 19115-19139 com del CSW ebRIM, creant problemes d'interoperabilitat que poden ser superats per un programari intermedi que actuï com un traductor de protocol (*broker*); aquest element tradueix una petició d'usuari genèrica des d'una interfície de portal únic a cadascun dels dialectes particulars de catàleg (Pasqual *et al.*, 2009).

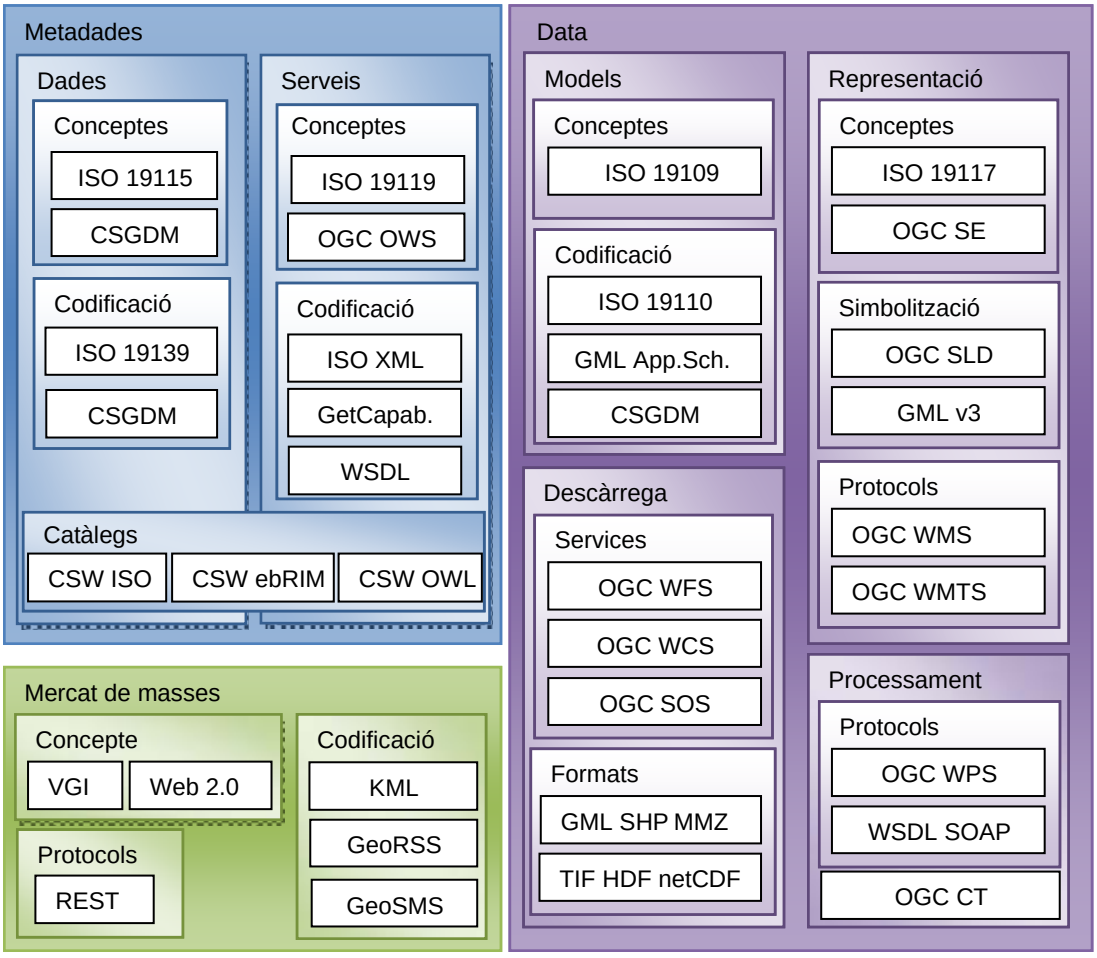


Figura 11: Estàndards potencialment utilitzats a les IDE classificats pel seu paper (font: figura 2, capítol 2).

Els identificadors de metadades i de dades s'utilitzen incorrectament o es confonen. Per això, cal fer servir identificadors únics i universals de metadades en lloc de *fileIdentifiers* locals, per tal de reconèixer els registres individuals, relacionar-los, i evitar que es presenti o es recatalogui el mateix registre de metadades més d'una vegada. Actualment, no existeix un consens sobre la forma de generar i gestionar els identificadors únics de metadades, i la versió actual de la ISO 19115 no inclou aquest concepte, sinó que ofereix un *dataURI* i un *MD_Identification* en la citació de dades. D'altra banda, no hi ha una forma satisfactòria de fer un enllaç bidireccional entre un conjunt de dades i el seu registre de metadades (Bernard *et al.*, 2005). A més, seguint els estàndards actuals de metadades és possible vincular les metadades a molts aspectes de les dades (figura 12), com per exemple una URL de descàrrega de l'arxiu, un servei de descàrrega, una descripció del model de dades, una llegenda, o una descripció de la codificació de la simbolització, entre d'altres. Aquests enllaços s'utilitzen molt poc ja que no són part del nucli de metadades (*core metadata*) i s'han definit com a entrades opcionals, però són importants per a les iniciatives de *linked data* (Goodwin *et al.*, 2008) i a la web en general, ja que permeten als usuaris obtenir i utilitzar les dades que acaben de ser descobertes.

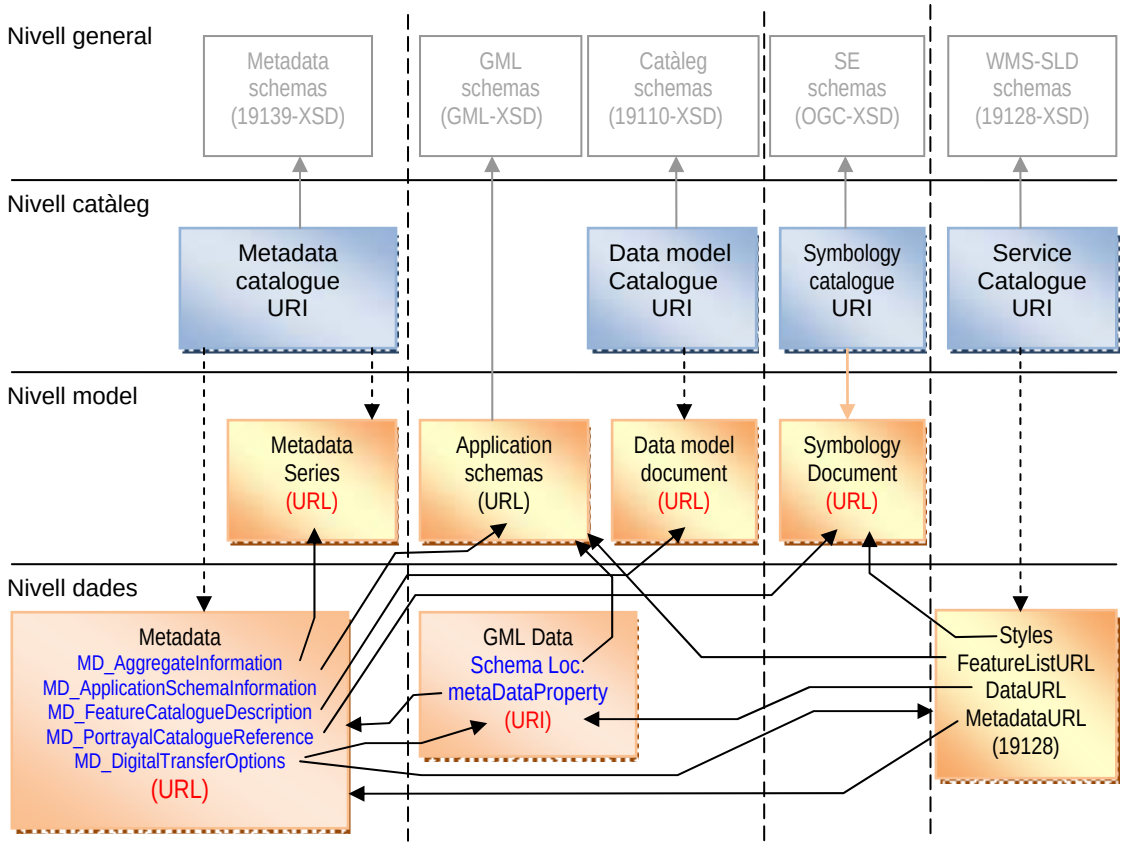


Figura 12: Relacions entre els diferents elements de les metadades sobre les dades, les metadades sobre serveis, i els estàndards de models de dades; a vegades realitzades per enllaços URL, a vegades per identificadors (font: elaboració pròpia).

Editar metadades manualment és un procés pesat de fer i propens a errors. Una nova investigació mostra que els diferents mètodes d'extracció automàtica de metadades (Han *et al.*, 2003) poden generar registres homogenis de metadades que estan ben sincronitzats amb dades en GML (Olfat *et al.*, 2010) o sobre una imatge, models digitals del terreny i altres formats vectorials (Manso *et al.*, 2009b), en lloc d'haver de ser elaborades en un procés totalment manual. A més, hi ha molt poques aplicacions que gestionin correctament les sèries cartogràfiques a partir de dades i metadades jeràrquiques, el que indica que cal més recerca sobre aquest tema (Zabala *et al.*, 2005).

La interoperabilitat semàntica segueix essent objecte de recerca i actualment és difícil d'aconseguir, però com a mínim cal potenciar l'ús de diccionaris de paraules clau com el tesaurus general multilingüe de medi ambient (*General Multilingual Environmental Thesaurus* [GEMET]) (Bernard *et al.*, 2005).

Millora de les metadades sobre serveis

Hi ha, com a mínim, tres maneres de codificar la descripció d'un servei: el document de descripció de metadades ISO 19119, el document de metadades de servei (ServiceMetadata) de l'OGC Web Services Common, i el Web Service Definition

Language (WSDL). Hi ha una necessitat d'harmonització i de tenir taules d'equivalències entre tots ells. També hi ha una clara necessitat de relacionar les metadades sobre serveis amb les metadades sobre les dades que contenen i a les quals proporcionen accés.

La informació que caracteritzi la disponibilitat i fiabilitat dels serveis cal que sigui mesurada provant els serveis de tant en tant, i els resultats s'han d'exposar junt amb les metadades del servei i amb estadístics de fiabilitat. També les mètriques sobre la intensitat d'ús dels serveis poden ser recollides i afegides a les metadades (Wang *et al.*, 2009).

Millora dels models de dades

A part de la secció 5 del CSDGM sobre les entitats i els atributs, hi ha dues altres formes genèriques de proporcionar descripcions dels models estandarditzats de dades: l'estàndard ISO 19110 (que especifica la forma de descriure els tipus d'entitats i atributs dels conjunt de dades) i l'estàndard ISO 19136-GML (que proporciona un conjunt de regles per a la descripció dels tipus d'entitat i atributs [propietats GML]). Les dues descripcions alternatives del model de dades es complementen entre elles i no s'han de veure com alternatives diferents. Hi ha la necessitat de més eines per a codificar i transformar descripcions de models de dades, encara que aquestes transformacions requereixin la intervenció humana per completar la feina (com en el programari Snowflake). Cal fer més recerca i desenvolupament en les transformacions del model de dades i els procediments de catalogació d'aquests.

Algunes propietats de les entitats contenen informació categòrica que pot prendre un conjunt de valors predefinits. Els diccionaris de categories (tesaurus) han de ser sempre distribuïts per alguna via; per exemple, podrien estar codificats en GML i vinculats a esquemes d'aplicació GML i les seves instàncies.

Millora de la descàrrega de dades

Les implementacions actuals de les IDE es centren clarament a proveir la representació de les dades en forma de mapa pictòric i no proporcionen suficient informació sobre el servei de dades equivalent (o és difícil de trobar) per permetre que les aplicacions i serveis pugin obtenir i analitzar les dades. Les eines d'anàlisi espacial necessiten tenir accés a les dades en si, però encara hi ha pocs serveis web de dades reals (per exemple, en WFS i WCS). De fet, Vandenbroucke (2008) va demostrar que a Europa, on molts països disposen de col·leccions completes de metadades i navegadors de mapes, hi ha només un petit percentatge de dades disponibles per a descarregar (amb excepció d'Àustria i Noruega). Per raons pràctiques, la segona generació de les IDE ha fracassat parcialment en aquest objectiu i s'ha introduït involuntàriament confusió en la disponibilitat de les dades a causa de, com a mínim, tres factors: en primer lloc, la disponibilitat de registres de metadades sembla ser suficient; en segon lloc, la indústria de serveis SIG ha implementat connectors WMS (Bernard *et al*., 2005) sense proporcionar capacitats de descàrrega; i la tercera, quan els proveïdors fan l'esforç per oferir a les dades, prefereixen fer-ho d'una forma estàndard, però quan es tracta de distribuir dades vectorials, resulta difícil decidir-se, perquè s'han d'enfrontar al format

GML, i només un conjunt molt limitat d'aplicacions pot llegir GML, a dia d'avui. Encara hi ha la necessitat de més eines per facilitar l'ús del GML. Mentrestant, hi ha un nombre relativament baix de formats que s'han convertit en estàndards *de facto* (SHP, etc) i hi ha col·leccions força completes de les eines de transformació de dades (entre aquests estàndards *de facto*) que funcionen raonablement bé, algunes d'elles de lliure accés, o fins i tot de codi obert (per exemple, http://www.gdal.org/).

Millora dels serveis de dades i addició de serveis de processament i de sensors

Els serveis web geospacials han de suportar un gran nombre de peticions concurrents, i necessiten fer càlculs complexos i requerint una gran capacitat de transmissió de dades (Tu *et al.*, 2004). De vegades és impossible de garantir la qualitat del servei i l'escalabilitat del disseny amb un sol ordinador i s'ha de considerar de fer servir un *cluster* d'ordinadors (Yang *et al.*, 2005). A la banda del client, aquests han de ser capaços de sol·licitar les dades de forma asincrònica en una estratègia multifil, el que permet als usuaris seguir interactuant amb el client en lloc d'haver d'esperar que la darrera petició respongui. És difícil trobar l'equilibri entre demanar només la part de les dades que l'usuari necessita realment (Scholten *et al.*, 2006), o reduir el nombre de sol·licituds simultànies al servidor, i anticipar-se a les necessitats de dades d'usuari mitjançant la transmissió de peces més grans d'informació (que potser mai no es faran servir en la seva totalitat). No obstant això, el nombre de peticions al servidor es pot reduir mitjançant l'ús dels mecanismes de memòria *caché* en servidors de mapes i de tessel·les (Yang *et al.*, 2005), o evitant que se sol·licitin entitats individuals i empaquetant convenientment les sol·licituds de col·leccions d'entitats (demanant les entitats d'un àmbit determinat de diversos tipus d'entitat, etc) a servidors WFS (Tu *et al.*, 2004). El capítol 6 avalua l'impacte de les sol·licituds simultànies en les implementacions actuals de servidors de mapes.

Alguns autors afirmen que els actuals serveis web de visualització i descàrrega són només un pas previ en lloc d'una solució definitiva, i pronostiquen que el geoprocessament és el següent pas necessari (Wright *et al.*, 2003). D'altra banda, el concepte emergent de la computació en el núvol (*cloud computing*) és vista com una oportunitat per ampliar les capacitats de processament distribuït. Els principals usos del geoprocessament que les IDE han fet fins ara són la transformació de coordenades i els serveis de transformació de format; però en realitat el WPS (Schut, 2007) podria encapsular interfícies per a gairebé tots els processos d'anàlisi espacial.

Mesurar les característiques físiques (per exemple, la temperatura) es pot aconseguir a partir de sensors fixos, o bé mitjançant sensors que van units a vehicles o bé transportats per persones i connectats a xarxes de sensors amb cable o sense fils (Craglia *et al.*, 2008). L'OGC ha anant desenvolupant un conjunt d'estàndards sota el nom genèric de Sensor Web Enablement (SWE), i en particular l'OGC Sensor Observation Service (SOS) (Na *et al.*, 2007) hauria de ser emprat per part de les IDE. És necessari harmonitzar les dades de sensors amb dades dels instituts de cartografia oficials, a partir de models de distribució, d'interpolació, etc, per tal de desenvolupar sistemes dinàmics capaços de proporcionar accés tant a les dades en brut de sensors, per als usuaris experts, com a dades més elaborades.

Millora la representació de dades i la simbolització

Per a poder incorporar, de manera que tingui sentit, les dades espacials de diferents fonts, es necessària una representació uniforme de les entitats espacials; tanmateix, sovint les organitzacions no estan disposades a modificar la seva presentació (o a donar facilitats per a modificar la seva presentació per part de tercers) de les dades espacials per adaptar-les a un altre propòsit, perquè consideren la seva simbolització com la seva marca d'estil. Per aquest motiu, la majoria de les implementacions de serveis de mapes no permeten el canvi dinàmic de la simbolització (Vandenbroucke, 2008). Un cop més, hi ha 3 maneres diferents de codificar la simbolització de dades: la ISO 19117, l'OGC Symbology Encoding (OGC-SE) i la codificació de la simbolització per defecte en GML v. 3.

Els navegadors web de mapes dels portals de les IDE han de proporcionar diversos preregistres temàtics preestablerts com a combinacions de mapes per mostrar la diversitat temàtica de les combinacions amb les dades disponibles. L'OGC Web Map Context (WMC) (Sonet, 2005) és un recurs tecnològic excel·lent per a la definició d'aquest registres preestablerts, ja que defineix una manera de guardar i recuperar les combinacions de capes WMS (de fet, és un recurs ja explotat fa prop de dues dècades per formats com l'MMM del MiraMon o es projectes de l'ArcView). El GetFeatureInfo del WMS i el WMTS o l'operació GetFeature s'han d'utilitzar més per a mostrar les dades alfanumèriques relacionades amb els mapes mostrats (Vanmeulebrouk *et al.*, 2009).

7.A.1.1.2. Fent Evolucionar el concepte de l'hipermapa: L'hipermapa mundial.

Les principals limitacions del vell concepte de l'hipermapa, introduït a la secció 1.A.2.1.3, són les següents: totes les consultes han de ser prèviament definides i "preparades", els enllaços es fan amb identificadors locals, els enllaços no defineixen una semàntica pel seu propòsit, i els usuaris només poden recuperar recursos. Hi ha quatre tecnologies que poden ser aplicades de forma conjunta en un sol sistema per tal de superar aquestes limitacions, com són, respectivament: els serveis geospacials web i els enllaços generats dinàmicament, l'ús de geo-identificadors globals, l'afegit del propòsit als enllaços i l'adopció de serveis RESTful per afegir les capacitats de crear, eliminar i modificar recursos. Quan se superen aquestes limitacions es pot parlar d'un sol hipermapa mundial (*World Wide Hypermap* [WWH]) amb capacitats millorades.

Si ens imaginéssim una col·lecció permanent de conjunts de dades (amb tots els seus components), que representés un món immutable que considera tots els aspectes imaginables, és fàcil generar un conjunt d'identificadors globals d'una manera centralitzada, i després codificar els vincles lògics entre ells. Però en el món real, els conjunts de dades es generen de forma progressiva i s'actualitzen periòdicament. Un conjunt de dades s'actualitza mitjançant la creació de noves entitats o modificant algunes de les anteriors. El capítol 3 proposa un protocol unificat per a l'administració de tots aquests recursos i un mecanisme per a generar i mantenir un sistema harmonitzat però descentralitzat global d'identificadors geospacials. La solució combina un nou servei web RESTful i aplicacions clàssiques de SIG lleugerament

modificades per tal de generar un WWH sense fissures en què els processos comuns de SIG es poden realitzar directament emprant una aplicació SIG clàssica però també en un entorn distribuït a la web. Instàncies locals del SIG poden accedir a les dades directament, mentre que els usuaris externs d'Internet poden també accedir-hi si fan peticions des d'una aplicació RESTful.

Els identificadors globals proposats són les URI de l'HTTP que es generen tan bon punt el recurs és creat. També es mantenen durant tot el cicle vital d'aquests recursos, fins i tot quan els elements es creen quan l'equip local on el SIG resideix està fora de la xarxa. Això garanteix la unicitat dels identificadors globals i evita la presència d'enllaços a recursos inexistents. Per fer això possible en un entorn descentralitzat, l'estructura de les URI es divideix en nivells o espais de noms. El primer nivell de la URI identifica l'organització i està formada per l'URI actual del lloc web principal seguida per la paraula "WWH" (usant el llenguatge de plantilles URI (Gregori, 2008, Gregorio, 2012) això s'expressa com: http://{OrganizationServer}/WWH). El següent nivell és el nom assignat a la instància del programari que s'utilitza per generar les dades (expressat com http://{OrganizationServer}/WWH/{SoftwareInstanceName}). El tercer nivell és el nom del conjunt d'entitats geospacials (per exemple, una col·lecció de cobertores o d'entitats) (expressat com: http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}. Els següents nivells poden identificar les entitats particulars, cobertores o fins i tot atributs de les entitats geospacials i així successivament, tal com s'indica a la taula 1 del capítol 3.

La unicitat de la URI completa depèn de 3 peces independents de programari que gestionen els nivells primer i segon, així com, els altres nivells més profunds (figura 13). La unicitat del primer nivell és proporcionada per les autoritats i mecanismes de noms HTTP (principalment els registres DNS). La unicitat del segon nivell és proporcionada per l'aplicació centralitzada RESTful dins el domini corporatiu. La unicitat dels següents nivells és proporcionada per cada instància de programari SIG.

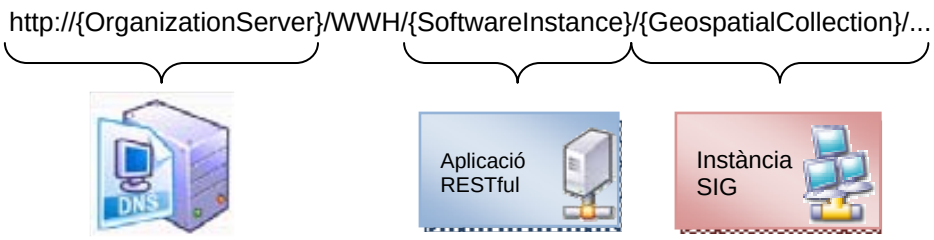


Figura 13: Entitat responsable de mantenir la unicitat de cadascun dels 3 nivells d'una URI de recurs dins del WWH (font: elaboració pròpia).

Cada instància de programari de SIG a la corporació pot gestionar els seus propis recursos (més detalls a la figura 14 i a l'Annex II). Els recursos administrats són col·leccions geospacials (col·leccions d'entitats, llistes de ràsters o bases de dades alfanumèriques), entitats geospacials (entitats, cel·les o registres) i els atributs complexos. Un cop identificats, aquests elements poden ser hipervinculats a recursos interns o externs. En el model del WWH, un hipervincle és una relació unidireccional

entre dos recursos del WWH. Els enllaços utilitzen identificadors globals que poden incloure el propòsit d'aquesta relació.

Seguint aquest esquema, les instàncies de programari de SIG són capaces d'interactuar directament amb els seus recursos de la forma habitual (accedint a l'arxiu binari, consultant la geodatabase, etc) i també són capaces d'interactuar amb les entitats remotes utilitzant peticions REST a l'aplicació RESTful corporativa corresponent emprant un dels quatre mètodes HTTP i un identificador global de recurs. Els quatre mètodes de l'HTTP en l'arquitectura REST (POST, GET, PUT i DELETE) s'associen amb les quatre accions comunes de qualsevol sistema informàtic (crear, recuperar, actualitzar i eliminar) per a cada tipus de recurs.

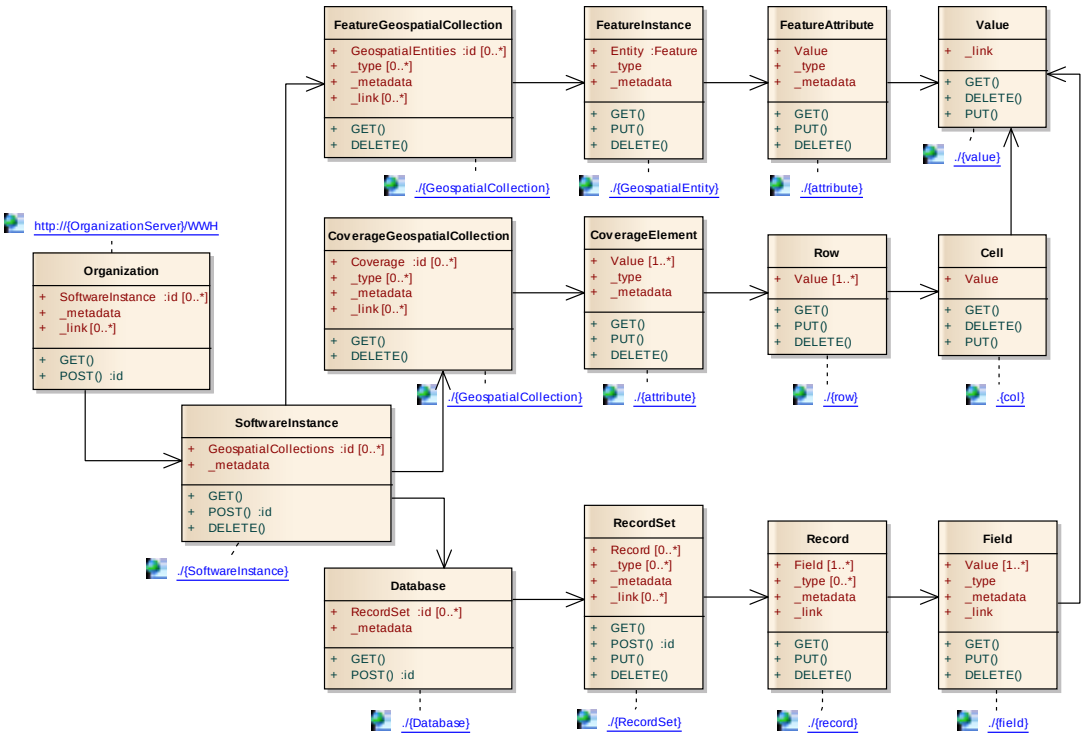


Figura 14: Resum dels recursos i les seves operacions, les relacions i les plantilles URI en el WWH (font: figura 2, capítol 3).

Hi ha cinc recursos especials que poden ser utilitzats afegint les paraules reservades "_metadata", "_type", "_link", "_all" and "_history" a un recurs donat. El significat es descriu en detall en el capítol 3.

A part de la recuperació d'una sola entitat, una operació molt comuna amb les col·leccions de d'entitats és l'extracció d'un subconjunt d'elles d'acord amb uns criteris espacials i/o temàtics. Aquesta funcionalitat es pot aconseguir en RESTful mitjançant la codificació amb una petició POST que s'envia a la URI de la col·lecció geospacial però acabada amb les paraules "_SpatialSubset" i/o "_FilteredSubset" incloent en el cos del missatge una descripció de l'objecte espacial codificada, per exemple, en GML, o una referència a ella (és a dir, un referència directa o una petició a un servidor WFS de l'OGC que recuperi un objecte a través normalment d'un requadre rectangular). El

resultat retornat és la URI d'una nova col·lecció geospacial que pot ser recuperada més tard, com qualsevol altra.

Catàlegs de la WWH

La solució proposada té diversos avantatges sobre un enfocament clàssic basat en l'estil d'arquitectura orientada a serveis. S'elimina la necessitat de serveis web específics d'accés a dades, així com haver de tenir diferents identificadors per a les dades, metadades i serveis d'accés a dades. En realitat, mitjançant un identificador únic i un conjunt RESTful de regles simples, els usuaris poden indicar les dades, obtenir les seves metadades i tenir accés a les dades reals. El WWH també fa que sigui completament innecessari catalogar serveis d'accés a dades. Els catàlegs de metadades CSW es poden utilitzar per catalogar i cercar metadades sobre les dades, però també s'espera que els cercadors web de text pla i els rastrejadors (*crawlers*) puguin ser útils pel descobriment de dades en el WWH.

D'altra banda, els catàlegs poden confiar en la unicitat de la URI del recurs original i, per tant, no hi ha necessitat de generar identificadors locals propis. La generació d'identificadors diferents per al mateix recurs és un problema que afecta molts catàlegs actuals (Nebert, 2004). Un altre avantatge és que els catàlegs de metadades poden usar la URI de dades com l'identificador de base per a les respostes del catàleg, proporcionant així un enllaç a les dades reals que els usuaris poden fer servir immediatament.

Serveis web al WWH

Tenir un identificador per a cada recurs geospacial des del moment en que es crea el recurs proporciona l'accés automàtic a qualsevol recurs geospacial i, per tant, no hi ha necessitat de serveis web de cobertores (Web *Coverage Services*) o d'entitats (Web *Feature Services*). Com que el servei d'accés a dades es implícit en el WWH, tampoc hi ha necessitat de serveis de catàleg d'accés a dades. Per tant, aquesta solució substitueix els actuals protocols WFS, WFS transaccional (WFS-T) i Web Coverage Service (WCS), però en canvi fa ús de diverses codificacions de l'OGC per intercanviar informació. De fet, les codificacions són necessàries per a comunicar representacions de recursos i el GML, el KML, el GeoRSS (Wick *et al.*, 2007) i el llenguatge de programació dels filtres (*Filter Encoding*) són excel·lents codificacions que es recomanen per a les representacions geospacials al WWH. No obstant això, els serveis de mapes, incloent els basats en tessel·les, poden ser oferts per cada col·lecció geospacial directament a través dels protocols WMS (de la Beaujardiere, 2004) o WMTS (Masó *et al.*, 2010a), considerant la URI de la col·lecció geospacial com l'arrel d'un servei WMS i afegint les corresponents parelles clau - valor (KVP) (KVP o REST pel cas del WMTS).

Finalment, l'aplicació RESTful pot ser ampliada per a proporcionar suport a l'execució remota de processos amb dades internes o externes. Els processos poden ser executats directament per l'aplicació SIG o es poden executar enviant un POST a la URI http://{OrganizationServer}/WWH/_SpatialOperation amb un document d'execució codificat tal com indica el Web Processing Service (WPS) (Schut, 2007).

Poseu trobar una llista completa dels recursos i les operacions definides sobre la idea del WWH a l'Annex II d'aquesta tesi doctoral.

7.A.1.2. Casos d'aplicació

En aquesta secció es presenten dues implementacions realitzades a partir d'aplicar els resultats conceptuals anteriors. La primera es troba descrita al final del capítol 2 mentre que la segona es troba descrita al final del capítol 3

7.A.1.2.1. La web *2.0 i els comentaris dels usuaris sobre les metadades: IDECTalk*

Les aproximacions estàndard actuals per a la captura de metadades requereixen una intervenció humana considerable, i són difícils de mantenir actualitzades. La seva principal funció és la de representar el punt de vista del productor de les dades (per exemple, sobre la qualitat i utilitat de les dades). Atès que la qualitat es defineix actualment com l'adequació al propòsit, els comentaris dels usuaris són essencials per a documentar les mesures i opinions que aquests puguin tenir sobre els conjunts de dades.

Algunes de les limitacions actuals de les metadades es podrien superar amb mètodes més participatius que permetin qualificacions i comentaris dels usuaris, com ja és una pràctica comuna en serveis comercials web 2.0. En aquest cas d'aplicació, posem en pràctica alguns dels aspectes discutits anteriorment a la secció "Adding Mass Marked, VGI and web 2.0" del capítol 2. En concret, es presenta un projecte pilot que permet la participació voluntària (VGI) sobre les metadades i demostra que alguns d'aquests problemes (que també afecten la IDEC) es poden superar amb un portal de cerca connectat amb el catàleg de metadades de la IDE. De fet, aquest portal permet demanar les metadades sobre un tema en particular i identificar un conjunt de dades concret. L'usuari pot llegir les metadades que el productor ha catalogat i també els comentaris anteriors dels usuaris. D'altra banda, pot entrar nous comentaris o actualitzar els seus. Tot això s'emmagatzema i queda immediatament a disposició dels altres usuaris.

L'entorn es basa en *mash-ups* sobre el catàleg de la IDEC i eines geospacials del mercat de masses (figura 15). Es tracta d'una aplicació CGI connectada a una base de dades que emmagatzema els comentaris dels usuaris i que resideix en un servidor web. Es fan servir els identificadors universals i únics (UUID) que la IDEC assigna a cada registre de metadades per vincular cada registre de metadades oficials amb el registre de comentaris dels usuaris. L'àmbit del conjunt de dades que el registre de metadades representa també es mostra usant l'API de Google Maps, com un altre *mash-up* (Chow, 2008).

L'entorn experimental utilitza deliberadament un estil mot similar als motors de cerca web i les aplicacions *blog* web perquè l'usuari se senti immediatament familiaritzat amb el sistema. Aquest entorn experimental es va desenvolupar en menys d'una setmana i sense comunicació directa amb el centre de suport de la IDEC. Això demostra que la creació d'aplicacions web 2.0 que tenen en compte els comentaris

dels usuaris connectats a un catàleg de metadades d'una IDE poden fàcilment ser desplegats per un tercer, que immediatament es converteix en part de la infraestructura de la IDE.

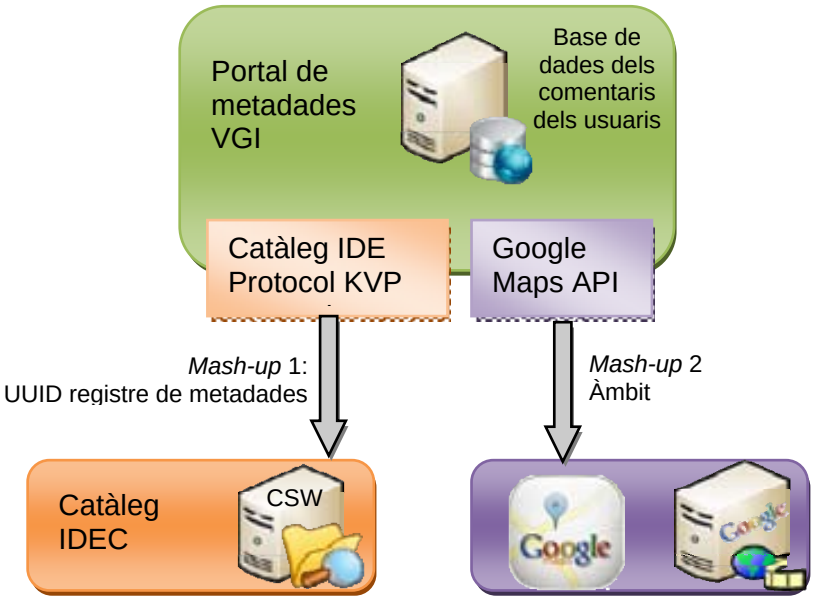


Figura 15: Arquitectura de l'IDECTalk. El *mash-up* 1 es connecta a la IDEC amb un UUID i el *mash-up* 2 es connecta amb el Google Maps utilitzant l'envolupant (font: figura basada en la figura 5 del capítol 2).

És clar que hi ha algunes qüestions que s'han d'abordar amb més profunditat, com les motivacions reals de les persones que proporcionen informació voluntàriament i el procés necessari per assegurar la qualitat de la informació proporcionada (Craglia *et al.*, 2008). No obstant això, creiem que l'experiment és un punt de partida interessant. Futures línies d'extensió d'aquesta plataforma són la introducció de funcionalitats per introduir paraules clau (*tags*) (Kalantari *et al.*, 2010) així com recollir dades sobre el comportament dels usuaris i afegir aquesta informació al sistema.

7.A.1.2.2. MiraMon-REST

La implementació del WWH requereix de programari addicional que pugui fer peticions sobre el protocol HTTP. Per tal de tenir una implementació de referència, discutim aquí com es pot implementar en una versió ampliada del programari MiraMon. El resultat és una implementació basada en el sistema d'arxius de disc d'aquest programari, però és possible implementar-ho també utilitzant altres sistemes de dades, com ara una base de dades geogràfica (*geodatabase*).

L'extensió del programari MiraMon pot ser explicada com una sèrie de modificacions al programari SIG MiraMon Professional i l'extensió del Servidor de Mapes del MiraMon (MMS) a un servei RESTful (figura 16). Les dues aplicacions comparteixen un nucli comú de funcionalitats, però també tenen algunes particularitats. Al nucli comú, ambdues aplicacions són capaces de realitzar el manteniment intern dels arxius de dades que els pertoquin directament i ambdues poden actuar com a clients RESTful

utilitzant HTTP. Són capaces de fer peticions com les que es presenten a la secció 7.A.1.1.2 i s'inclouen en el capítol 3 (llistats a la taula 1 i 2, i explicats a la secció 3).

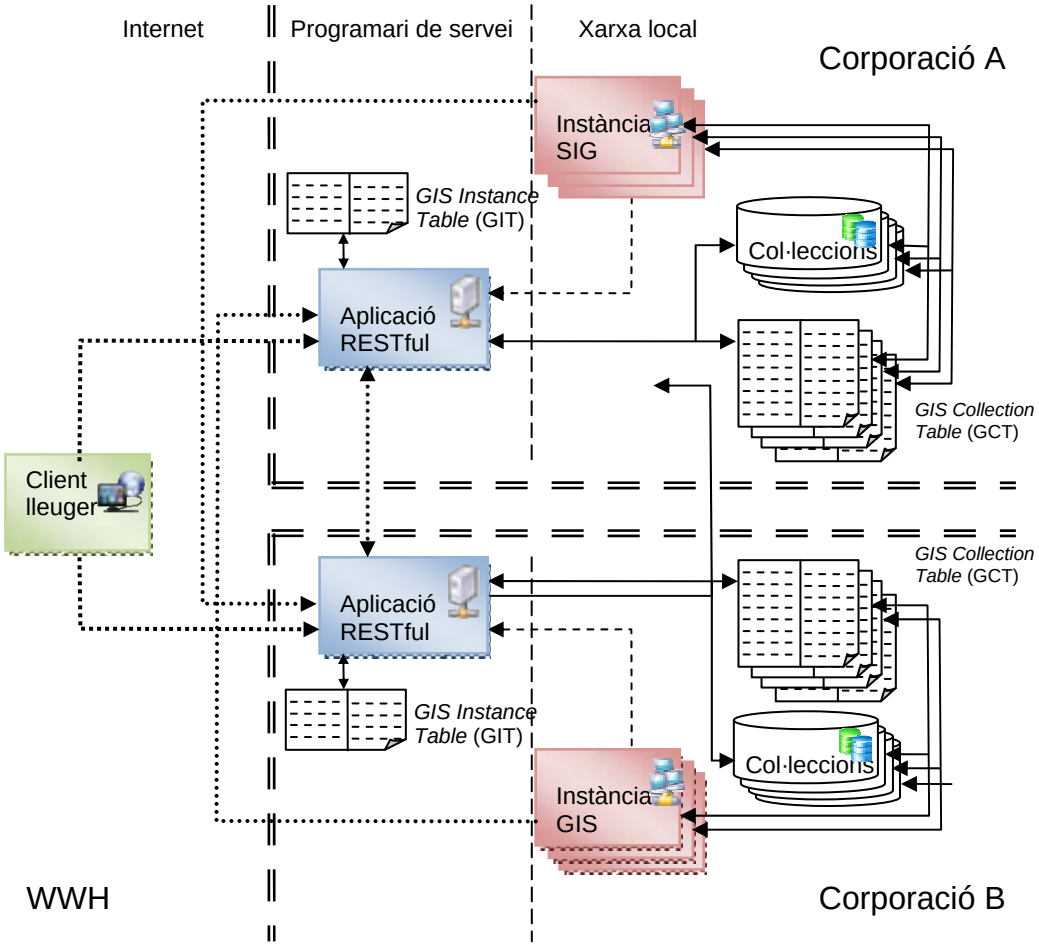


Figura 16: Components de programari i les connexions entre 2 nodes corporatius i un client lleuger extern en el WWH (font: figura 3, capítol 3).

Modificacions en el component intern del SIG

Les principals funcionalitats del programari MiraMon no necessiten ser modificades, però s'ha de fer diversos afegits. El model de dades d'arxiu cal que sigui modificat (mantenint la compatibilitat descendent) per permetre guardar l'identificador de les col·leccions (registrat com a un camp de metadades), i els enllaços a altres col·leccions geospacials (registrats com a camps de metadades) i a les entitats geospacials i atributs (emmagatzemats com atributs de base de dades). També cal afegir el suport a les referències a entitats geospacials externes. A més, també cal afegir la capacitat de funcionar com un client RESTful (i fer una petició a qualsevol altre servei RESTful del WWH).

El programari MiraMon d'escriptori pot utilitzar la metodologia descrita prèviament per accedir a la URI de la llista de col·leccions geospacials disponibles. Les col·leccions geospacials es transfereixen i es mostren llavors a l'usuari. Una altra modificació important és l'addició de la taula de col·leccions geospacials (*Geospatial Collection Table* [GCT]) que manté un registre dels identificadors de totes les col·leccions

(conjunts de dades) mantinguts per aquesta instància SIG particular, i el mètode d'accés necessari per a la gestió de les dades (per exemple, una ruta d'arxiu o els paràmetres d'accés a la geodatabase).

Servidor RESTful corporatiu

Aquesta aplicació és una extensió de la CGI MMS capaç de respondre a les peticions RESTful. Cal reconèixer els quatre mètodes HTTP i traduir-los a accions que s'han de dur a terme sobre els recursos identificats. D'aquesta manera l'entorn és capaç de reconèixer un identificador de recurs global, de buscar la informació geogràfica que guarda les dades, de descobrir on està emmagatzemat el recurs real i dur a terme l'operació sol·licitada directament sobre les dades. Per fer això possible, es manté una llista de totes les instàncies SIG a la corporació (en una taula d'instàncies SIG, *GIS Instance Table* [GIT]). Quan una instància de SIG nova s'instal·la en un equip intern concret, també crea la seva pròpia GCT. A més, es crea un identificador de recurs amb una operació POST al http://{OrganizationServer}/WWH, enviant informació sobre com accedir a la GCT. Això fa que el servei RESTful registri la instància SIG a la GIT.

Quan una aplicació RESTful corporativa rep una sol·licitud d'informació interna, en rep l'identificador global. Aquest identificador es pot separar en 3 nivells. El primer nivell s'utilitza per identificar la sol·licitud com una URI a un recurs intern i acceptar la sol·licitud. El segon nivell permet identificar la instància de programari i llegir la GIT per determinar la forma d'accedir a la taula d'instàncies SIG (GCT). El tercer nivell ofereix informació sobre la col·lecció geospacial sol·licitada; a la GCT es troba la informació de com accedir directament a les dades reals. L'aplicació RESTful també pot actuar com un programari SIG perquè té un identificador de programari reservat, http://{OrganizationServer}/WWH/_external, i també administra la seva pròpia GCT per a les col·leccions que es creen des de l'exterior sense la intervenció de cap tipus de programari de SIG intern, com per exemple les sol·licituds de filtres espacials, filtres temàtics o els resultats del processos, així com el simple enviament de noves dades amb una operació POST.

7.A.2. Aspectes específics de la navegació de mapes a Internet

La recerca realitzada sobre els navegador de mapes a Internet ha conduit a un conjunt de resultats que es resumeixen en aquesta secció i que han estat majoritàriament extrets i tractats en els capítols 4, 5 i 6 d'aquesta tesi.

7.A.2.1. Aspectes conceptuals i d'arquitectura

En aquesta secció es resumeixen els resultats més conceptuals i d'arquitectura que contribueixen a millorar la interoperabilitat i el rendiment dels serveis de mapes. Per aconseguir això es proposa un nou estàndard i una implementació de serveis de mapes que utilitzi el JPEG2000.

7.A.2.1.1. Servei web de mapes tessel·lats (WMTS)

El WMTS (capítol 4) proporciona una solució basada en estàndards per servir mapes digitals que utilitza tessel·les predefinides. El servei publica l'esquema de tessel·les

adoptat i el fa avinent (anomenat TileMatrixSet, que està compost per una llista de les escales disponibles a les quals hi ha disponibles tessel·les i un patró de tessel·les regular per a cadascuna de les escales [Figura 17b)]) en el document de metadades de servei (ServiceMetadata), comú a tots els serveis web de l'OGC. Aquesta declaració indica les tessel·les disponibles per cada capa (és a dir, cada tipus de contingut), per cada estil de representació gràfica, per cada format, per cada sistema de referència, per cada escala i per cada fragment geogràfic de l'àrea total coberta. A més a més, el servei també pot oferir la possibilitat (opcional) de retornar la informació d'un punt del mapa sobre la tessel·la en forma de descripció textual (FeatureInfo).

Aquest estàndard especifica els diferents mecanismes concrets d'intercanvi entre clients i servidors per a dos estils d'arquitectura diferents. L'estàndard defineix les operacions GetCapabilities, GetTile i l'operació opcional GetFeatureInfo per a l'estil d'arquitectura orientada a objecte basat en l'ús diversos sistemes de codificació de missatges, inclosos els missatges codificats usant parells clau-valor (KVP), missatges XML, o fitxers XML incrustats en missatges SOAP. L'estàndard també defineix els mecanismes de sol·licitud via l'estil d'arquitectura orientada a recursos basat en portes d'entrada URL; això permet als clients, demanar de forma simple el document de ServiceMetadata, les tessel·les i els documents FeatureInfo com a recursos. Finalment, fa algunes recomanacions per millorar la interoperabilitat entre les implementacions, com ara l'adopció dels conjunts d'escales ben coneguts (*Well Known Scale Sets* [WKSS]), que són conjunts d'escales predefinits al propi estàndard o per altres organismes reconeguts i utilitzats per un gran nombre de servidors, fet que facilitat la superposició de les seves capes en un sol client (figura 17a).

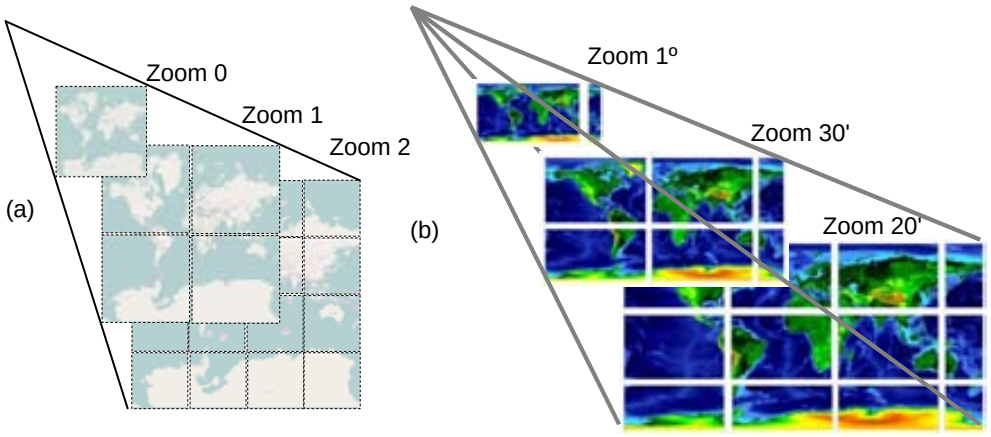


Figura 17: Dos TileMatrixSets diferents formats per diverses TileMatrix que utilitzen conjunts d'escales diferents (3 són visibles a la figura). (a) està usant els WKSS *GoogleMapsCompatibleWKSS* (les tessel·les provenen de l'OpenStreetMap); on cada TileMatrix es divideix en quadrats de mosaics regulars. (b) no segueix una WKSS i utilitza tessel·les rectangulars (font: elaboració pròpia).

Des de l'aprovació d'aquest estàndard, trobem diverses implementacions del mateix disponibles i han aparegut força publicacions científiques que fan referència a aquest document normatiu relativament nou i que es troba íntegre en el capítol 4.

7.A.2.1.2. Ús del format JPEG2000 en servidors de mapes

Per entendre aquesta discussió cal tenir en compte que un JPEG2000 pot definir una estructura de tessel·les només per a la resolució més detallada, però la compressió es realitza en l'espai *wavelet* transformat. Per tal d'optimitzar un arxiu JPEG2000 perquè pugui ser servit com WMTS, cal tallar-lo seguint el mateix patró que les tessel·les WMTS que s'exposaran al nivell de resolució de base. Un servidor WMTS basat en JPEG2000 ha de definir l'escala següent, amb un denominador d'escala que ha de ser el doble de l'anterior escala, amb el mateix origen (TopLeftCorner) i el mateix ample i alt en píxels (com a resultat tenim que cada tessel·la d'aquest nivell cobreix 4 tessel·les del nivell anterior ja que s'ha doblat la regió). Seguint aquest esquema, l'extracció d'una tessel·la WMTS a la resolució de base (de més detall) suposarà llegir una tessel·la JPEG2000 i les dades de tots els nivells interns de resolució del JPEG2000. L'extracció d'un tessel·la del segon nivell de l'escala WMTS requerirà la lectura de les dades de 4 tessel·les del JPEG2000 i accedir als *codeblocks* de tots els nivells de resolució JPEG2000, excepte el més detallat. L'extracció d'un tessel·la del tercer nivell d'escala requereix la lectura de les dades de 16 tessel·les JPEG2000 i accedir a les dades de tots *codeblocks* de tots els nivells de resolució, excepte els 2 de més detall, i així successivament (figura 18).

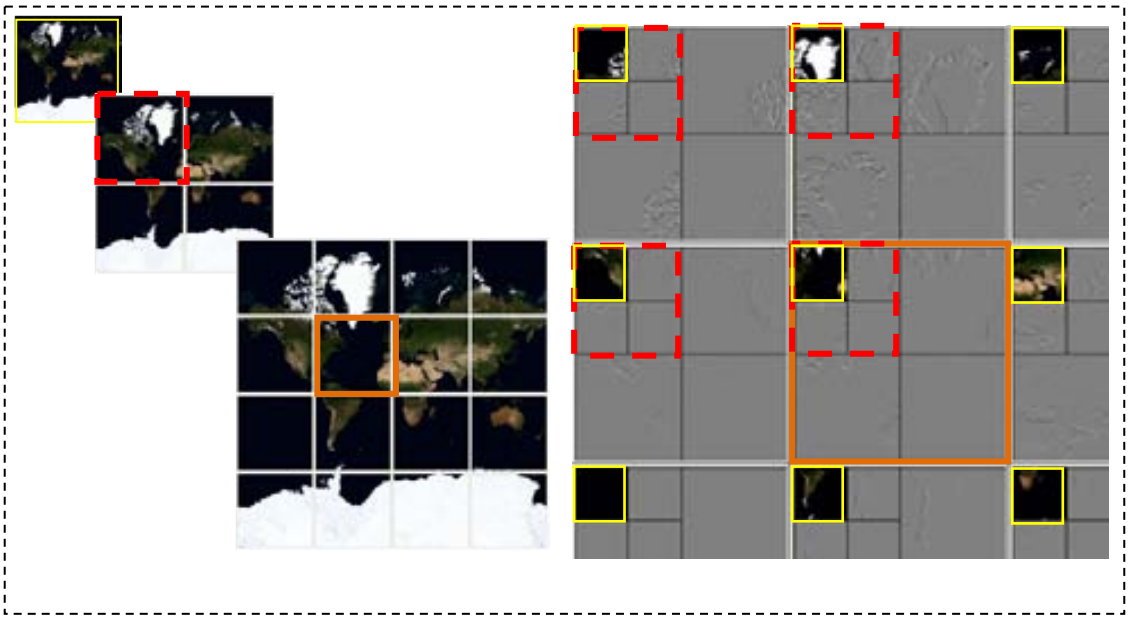


Figure 18: Tessel·les a 3 nivells d'escala i correspondència entre *codeblocks* a 3 nivells de resolució de la transformada *wavelet*: en color taronja es representen les equivalències en el nivell de resolució de base, en color vermell equivalències en el segon nivell de resolució, i en color groc les equivalències en el tercer nivell de resolució (incomplet per raons de llegibilitat) (font: elaboració pròpia).

En alguns casos, quan es necessita molts nivells de resolució WMTS o les imatges involucrades són molt grans, pot ser difícil d'obtenir els nivells de baixa resolució a partir d'un arxiu JPEG2000 complet, per culpa del nombre de tessel·les JPEG2000 involucrades. Per aquesta raó es recomana generar algunes imatges independents (2, 3, ...) de resolucions encadenades i combinar-les en una sola capa WMTS. L'arxiu JPEG2000 que s'utilitzarà en un WMTS tindrà un rendiment molt millor si es prepara

(comprimeix) en tessel·les d'una mida petita, com les típicament usades en serveis WMTS (és a dir, 256x256 o 512x512).

Aquesta estructura de nivells s'exposa en la descripció de la capa del document de metadades del servei WMTS com un nivell d'escala bàsic. Els següents nivells de resolució de la capa compartiran el mateix origen i mides fins arribar al mateix nombre de nivells de resolució que hi ha dins dels arxius JPEG2000 (el client no serà conscient que internament hi hagi diversos d'arxius d'entrada). Un servidor WMTS que rebi una sol·licitud WMTS haurà de traduir-la en ordres de la rutina de descompressió JPEG2000 (figura 19).

Comparant WMTS i serveis JPIP.

En termes generals, el protocol JPIP podria ser considerat una alternativa al WMTS ja que ambdós comparteixen l'interès comú de proporcionar un protocol que permeti la difusió d'imatges a través de la xarxa d'una manera ràpida i eficient. Hi ha 2 principals diferències entre el WMTS i el JPIP: la primera recau en el fet que el WMTS normalment utilitza formats comuns a Internet (PNG, JPEG, etc) i pot utilitzar eventualment JPEG2000, però el JPIP només pot utilitzar JPEG2000. Això permet que el WMTS pugui ser utilitzat en un navegador web actual sense necessitat de cap programari addicional i sigui ideal per combinar amb altres aplicacions web. Per contra, el JPIP requereix un client més especialitzat que inclogui un programari addicional que permeti la lectura de JPEG2000. La segona diferència és que el protocol JPIP permet mantenir la *caché* de *codeblocks* en el client i en el servidor, de manera que els *codeblocks* rebuts en transmissions anteriors no han de ser retransmesos de nou. Aquesta propietat permet estalviar temps i ample de banda quan es demana un nivell de zoom (resolució) diferent d'una regió de la imatge que s'havia sol·licitat prèviament a una altra resolució, però per contra requereix que el servidor recordi l'estat de cada client, ja sigui mantenint una sessió al servidor o mitjançant la transmissió de l'estat del client al servidor a cada petició. L'especificació WMTS actual no pot incloure això i, en les mateixes circumstàncies, les implementacions WMTS han de transferir informació redundant. No obstant això, el WMTS està completament dissenyat com un protocol sense estat, de manera que pugui aprofitar els mecanismes d'emmagatzematge *caché* de la web que finalment beneficiaran a tots els usuaris WMTS concurrents d'un servidor en particular; en conseqüència el WMTS és més apropiat que el JPIP per a serveis web de mapes molt populars. A més, el WMTS té una interfície RESTful que permet desplegar un servei WMTS en un servidor web estàndard (per exemple, Apache, IIS, etc) com un conjunt d'arxius estàtics prerasteritzats sense necessitat de cap programari addicional (fent que el procés de desplegament sigui compatible amb tots els proveïdors de serveis d'Internet actuals).

Les futures implementacions de l'estàndard WMTS podrien ser esteses de manera que s'eliminessin algunes transmissions redundants si un client WMTS ha rebut prèviament una tessel·la de la mateixa regió d'una resolució més grollera sense trencar la compatibilitat amb l'especificació WMTS actual. Una possible tècnica consistiria en la transmissió de la tessel·la de la resolució més grollera i després, quan se sol·liciti una resolució més detallada, només s'enviï la informació necessària per enriquir la tessel·la prèvia a aquest nou nivell de detall. Això es pot aconseguir mitjançant l'ús d'una

piràmide laplaciana (Burt *et al.*, 1983). En lloc d'emmagatzemar imatges completes veritables per a cada nivell de la piràmide (com fa la piràmide gaussiana), en una piràmide laplaciana, la informació emmagatzemada per a cada nivell particular, es construeix com la diferència entre la imatge gaussiana d'aquest nivell i la imatge gaussiana del següent nivell de resolució menor. Tenint la imatge més grollera i la informació de la laplaciana per al següent nivell, és possible reconstruir la imatge real per al següent nivell de resolució de la piràmide. No obstant això, un servei WMTS hipotètic que proporcioni nivells de resolució laplacians, requerirà un client més pesat i especialitzat que emmagatzemi els nivells de resolucions anteriors més grollers i que combini la informació dels següents nivells per mostrar la imatge real. Aquest enfocament té una sèrie de requeriments similars als d'un client JPIP però no proporciona les funcionalitats addicionals al protocol JPIP com és una millor compressió i una més gran flexibilitat d'accés espacial aleatòria (Patel *et al.*, 2007), però seguirà sent un solució sense estat (*stateless*) que permet un ús correcte dels mecanismes *caché* d'Internet i tenir una millor escalabilitat en casos de molts usuaris concurrents.

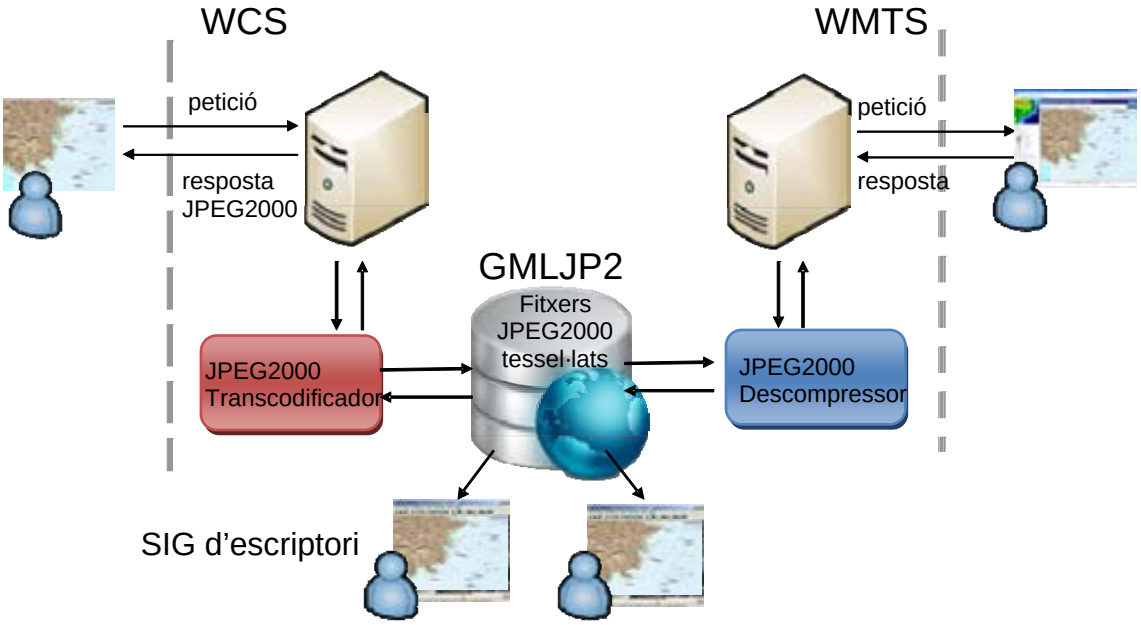


Figura 19: Un sol repositori de JPEG2000 internament en tessel·les (amb les metadades codificades usant GMLJP2) pot server per SIG interns d'escriptori, per a servidors WCS i per a servidors WMTS (font: figura 9, capítol 5).

Finalment, hi ha una altra raó per a utilitzar el format JPEG2000: Una corporació pot tenir un magatzem de dades com un repositori únic d'arxius JPEG2000 que es poden fer servir com a dipòsit intern per els serveis web WMTS, WCS i SOS i, alhora, utilitzar-los directament en les aplicacions de teledetecció i SIG corporatiu, evitant completament la duplicitat de formats de dades i proporcionant un format comprimit d'emmagatzematge d'alta eficiència (figura 19).

7.A.2.2. Casos d'aplicació

Aquesta secció resumeix els resultats de l'anàlisi detallada del rendiment de servidors (WMS i algunes estratègies basades en tessel·les, entre elles el WMTS exposat en el capítol 4). Els resultats d'aquesta secció justifiquen l'aproximació de l'estàndard WMTS i exposen com afecten al rendiment les diferents estratègies utilitzades pels diferents fabricants. Entre els fabricants analitzats destaquem l'Express Server de Lizardtech que utilitza un format basat en *wavelets* molt similar al JPEG2000, analitzat a la secció 7.A.2.1.2 i al capítol 5.

7.A.2.2.1. Anàlisi de rendiment dels servidors del mapes

En aquesta secció s'exposa l'avaluació realitzada als set servidors que considerem més populars (el GeoServer, el MapServer, el Servidor de Mapes del MiraMon, l'Express Server, l'ArcGIS Server, el TileCache i el GeoWebCache) que fan servir estàndards de serveis de mapes (WMS, WMTS, WMS-C, TMS), i es determina els seus temps de resposta, les funcionalitats i la facilitat d'ús de cadascun dels servidors. A més, les diferents aproximacions de tessel·les en implementacions seqüencials, simultànies i semi concurrents reben una especial atenció.

Avaluació de les sol·licituds concurrents a un sol servidor

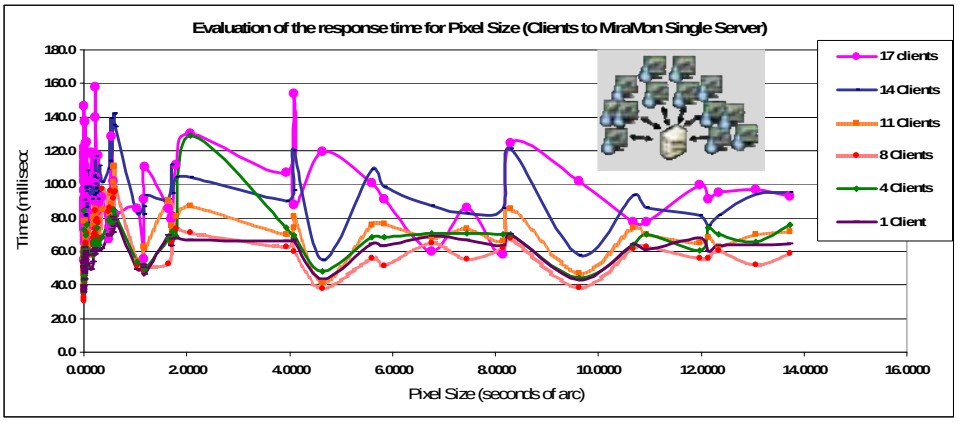
Un dels principals factors que afecten el rendiment dels servidors web és la concurrència en les sol·licituds. El capítol 6 mesura la influència de la mida de píxel (escala) i de la mida de la vista en els temps de resposta de les sol·licituds WMS. Més d'un centenar de sol·licituds diferents varen ser realitzades, totes fetes des d'un màxim de 6 clients simultanis. Un resultat immediat en totes les circumstàncies mesurades és l'augment del temps de resposta en augmentar el nombre de clients simultanis fent peticions a un únic servidor. El servidor més ràpid és l'Express Server, probablement a causa de la naturalesa del format comprimit *wavelet* (MrSID o JPEG2000) que s'organitza internament en una piràmide de nivells de zoom. L'ArcGIS Server és el servidor que utilitza les imatges originals en format GeoTIFF que té els millors resultats. En un rendiment intermedi trobem el servidor del MiraMon, que requereix d'un procés previ de preparació de les capes. Com que el MapServer està programat en llenguatge C, esperàvem millor rendiment davant del GeoServer, desenvolupat en Java. Aquesta intuïció es va confirmar amb peticions fetes des d'un sol client, però el GeoServer ha resultat més ràpid que el Mapserver davant de peticions concurrents. També s'ha trobat petites diferències en les temps de resposta, depenent del format de sortida sol·licitat, essent el més ràpid de generar el format JPEG en el MapServer, l'Express Server, l'ArcGIS Server i el servidor de Mapes del MiraMon, però el format PNG en el GeoServer.

Avaluació d'un *cluster* de servidors

En l'estat actual de maduresa del maquinari, no és possible augmentar dràsticament el rendiment a partir d'adquirir una màquina molt més ràpida, fins i tot estant disposat a pagar més. La tecnologia informàtica actual sembla haver arribat a un sostre de velocitat de processament de la CPU i la velocitat d'accés a disc. Per superar la

degradació del rendiment observat en les sol·licituds simultànies, una possible solució alternativa és la creació d'un *cluster* de servidors que poden actuar com un únic servidor virtual que s'ocupa de les sol·licituds en paral·lel. S'ha dut a terme proves de comparació d'un WMS com a un sol servidor i un WMS com un *cluster* d'ordinadors en configuració de balanceig de càrrega de xarxa (Network Load Balance[36] [NLB]), en el qual 6 equips responen coordinadament a diferents clients com si es tractés d'un sol servidor més ràpid. Aquestes proves s'han fet per un màxim de 17 clients simultanis, avaluant el temps de resposta pel cas del Servidor de Mapes del MiraMon.

(a)



(b)

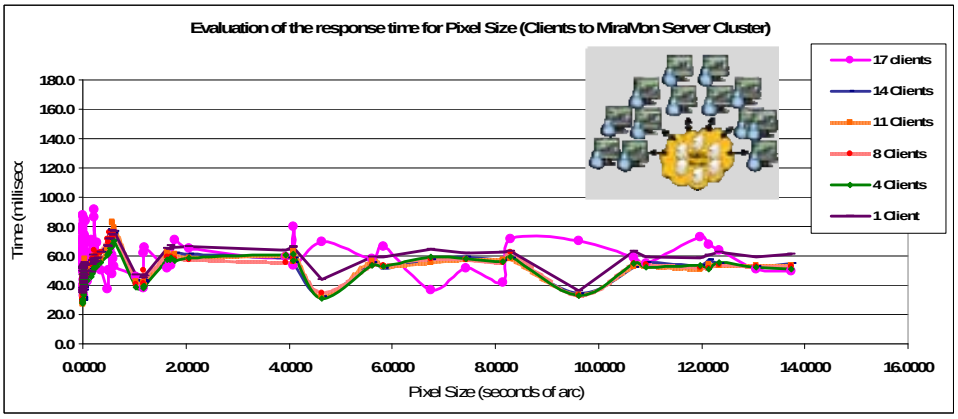


Figura 20: Temps de resposta de les distintes sol·licituds simultànies amb un màxim de 17 clients a un servidor WMS del MiraMon: (a) configuració de servidor únic (font: figura 4, capítol 6), (b) configuració en *cluster* de 6 servidors (font: figura 5, capítol 6).

Les mesures de temps de resposta comparant un únic servidor i el *cluster* de 6 servidors que reben peticions de múltiples clients revela que el temps de resposta del servidor NLB és molt més estable i gairebé equivalent al cas d'un sol client, fins i tot pel cas de 17 peticions simultànies (com es mostra a la figura 20). Esperem que hi hagi alguna degradació si s'augmenta encara més el nombre de peticions simultànies, però afortunadament, el rendiment del *cluster* NLB es pot millorar més, mitjançant l'agregació de més servidors al *cluster* (fins a 64 en Windows 2003). Si suposem que el

[36] NLB és una de les aproximacions més senzilles als *clusters* d'ordinadors on uns quans ordinadors comparteixen la mateixa adreça web i el mateix contingut. Quan una petició d'un client arriba, aquesta és assignada a un sol ordinador que la respondrà. La següent petició serà assignada a altre ordinador diferent.

rendiment és lineal, això significa que aquesta mateixa configuració es pot estendre com a mínim fins a unes 200 sol·licituds simultànies, sense degradació de rendiment. Tenint en compte que el temps de resposta per aquestes peticions és sempre menor a 0.1 segons, aquesta configuració podria resoldre l'equivalent a 2000 sol·licituds per segon sense cap degradació del rendiment.

Rendiment de peticions i respostes per tessel·les

En el capítol anterior hem estat usant la típica interacció WMS, on un client WMS sol·licita tota la imatge necessària per cobrir la vista del client en una sola peça (com s'explica a la figura 6). Recentment, alguns clients WMS (com l'OpenLayers) són capaços de dividir l'espai de la vista en una matriu regular de tessel·les. En fer això, es necessiten diverses tessel·les per cobrir tota la finestra de la vista, però per contra, el client pot reciclar algunes tessel·les quan l'usuari mou la vista lateralment i també pot treure avantatge dels mecanismes de *caché* d'Internet. No obstant això, aquesta estratègia pot tenir els seus inconvenients si els mecanismes de *caché* no poden ajudar i el servidor no s'ha preparat per gestionar aquesta situació, pel fet que la vista es composa per diverses tessel·les que són sol·licitades al mateix temps al servidor, i, com hem comentat anteriorment, els temps de resposta augmenta en el cas de rebre peticions simultànies, tot i que els usuaris podrien no tenir aquesta percepció pel fet que algunes tessel·les arriben al client abans i es mostren immediatament. Els resultats que s'exposen aquí clarifiquen els efectes de les peticions per tessel·les sobre servidors WMS clàssics i quantifiquen la diferència entre el lliurament ràpid d'una imatge completa i les diverses imatges de tessel·les d'una manera independent a la percepció de l'usuari. També s'ha analitzat les millores que s'experimenten quan el servidor també està optimitzat per servir tessel·les i empra un protocol destinat a aquest fet, com es: OSGeo WMS-C i OGC WMTS.

Hem simulat els clients de tessel·les (tessel·les de 256x256 píxels) que fan sol·licituds típiques WMS, que no tenen cap estratègia en particular per servir tessel·les, en les següents tres configuracions: WMS tessel·lat en un nombre il·limitat de sol·licituds simultànies per un sol client, WMS tessel·lat en peticions semi-concurrents en paquets de quatre tessel·les a la vegada i WMS tessel·lat en peticions en seqüència. També hem comparat aquestes tres configuracions amb l'ús clàssic del WMS en una sola imatge per tota la vista. Els resultats òptims depenen del producte analitzat, però les peticions WMS en una sola imatge i les sol·licituds semi-concurrents són les configuracions més ràpides.

També, els productes que han estat optimitzats per a respondre tessel·les han estat avaluats, ja sigui perquè proporcionen una forma de preparació prèvia tessel·les o una manera de guardar les tessel·les que es genera sobre la marxa per al seu ús posterior. El principal inconvenient és el temps de generació, però això pot ser millorat parcialment amb estratègies de metatessel·lat *(metatiling*). En lloc de generar cada tessel·la individualment, una metatessel·la *(metatile*) es genera creant un mapa de dimensions més grans que després és divideix en una matriu de tessel·les més petites. Per a tots els servidors analitzats, peticions a imatges de 512 x 512 necessiten més temps, però no tant de temps com la suma dels temps requerida per a generar les quatre imatges 256 x 256 equivalents seqüencialment.

7.B. Summary of results (English version)

The summary of results of this PhD thesis encompasses an overview of the outcomes obtained in the publications included as chapters 2 to 6. First we present the general aspects raised from the implementation of geospatial standards in spatial data infrastructures (SDI) and in Geographic Information Systems (GIS) in general, and later map browser specific aspects are covered.

7.B.1. Generic aspects derived from geospatial standards implementation

The research done on the degree of interoperability of SDI, systems of systems and technologies for the digital Earth has led to a set of results summarized in this section, mostly extracted and discussed in chapters 2 and 3 of this PhD thesis.

7.B.1.1. Conceptual and architectural aspects

This section summarizes the results of conceptual architecture and discusses the weaknesses of both, found in standards documents as well as those found in their implementations. Also, measures to improve this situation and improve geospatial interoperability as proposed.

7.B.1.1.1. Results from the spatial data Infrastructures analysis.

The presented results on this topic are a set of recommendations for improving SDI implementations, such as adding more functionalities, enhancing interoperability or improving performance. These recommendations have been classified into the following topics: Metadata about data and services, Data models, Data download, Data and Process Web services, Portrayal and symbolization and Mass market. The standards considered in this study are represented in Figure 11.

When providing these recommendations we consider the user side and the service side, in parallel. This summary collects and comments some of these recommendations, but Table 2 in chapter 2 contains a more complete set.

Improving metadata about data

Metadata catalogue standards come in a variety of versions and profiles of both CSW ISO 19115-19139 and CSW ebRIM creating interoperability problems that can be overcome by a middleware software that acts as a protocol translator, or *broker*, which translates a generic user request into each particular catalogue dialect from a single portal interface (Pasqual *et al.*, 2009).

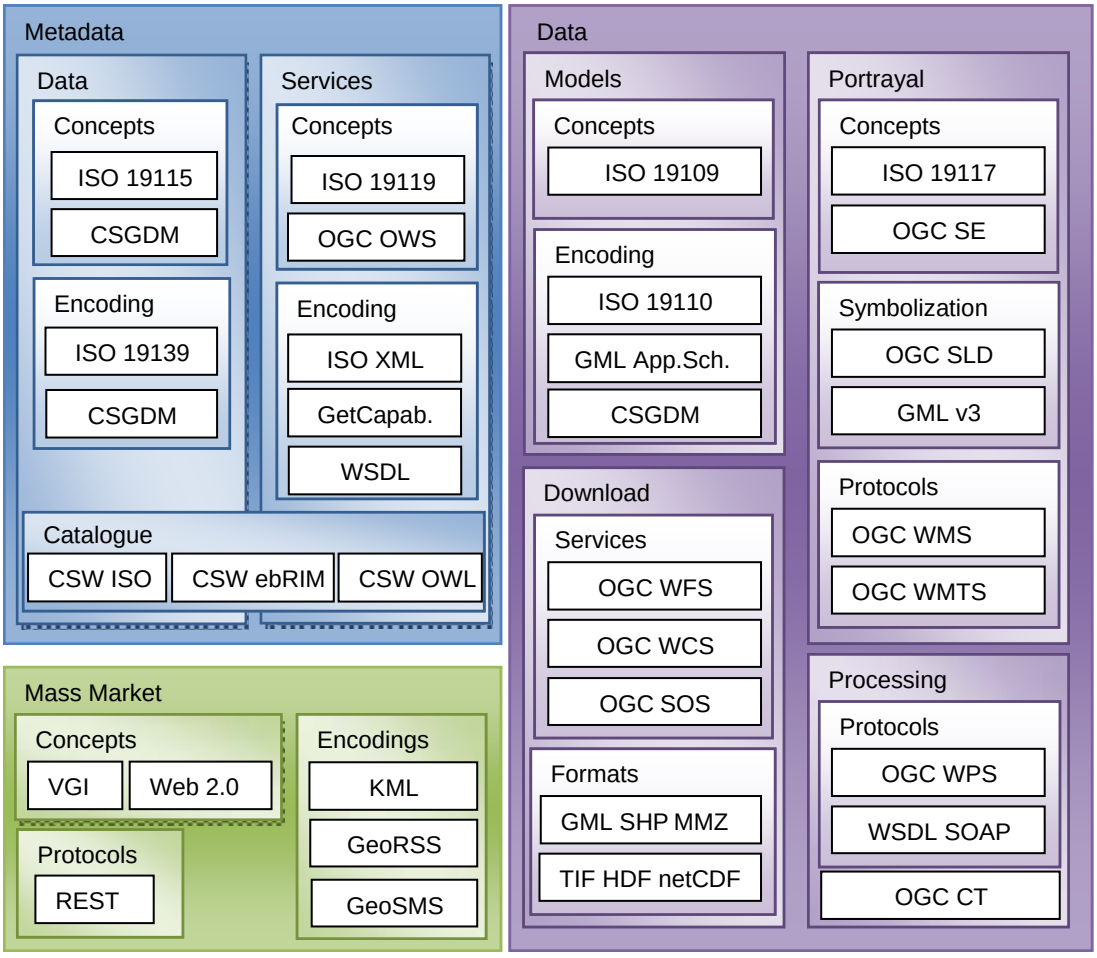


Figure 11: Standards potentially used in SDI classified by its role (source: Figure 2, chapter 2).

Metadata and data identifiers are used improperly or confused. Universal and unique metadata identifiers need to be used instead of local fileIdentifiers in order to recognize individual records, relate them, and to avoid presenting or reharvesting the same metadata record more than once. Currently, there is no consensus on how to generate and manage metadata unique identifiers, and the current version of ISO 19115 does not include this concept but rather provides a dataURI and a MD_Identificacion in the data citation. Furthermore, there is no satisfactory way of making a bidirectional link between a dataset and its metadata record (Bernard *et al.*, 2005). Additionally, current metadata standards can link to many aspects of the data (Figure 12), such as a download URL to a file, a download service, a data model description, a legend, or a symbolization encoding description, among others. These links are used very little as they are not part of the metadata core and have been defined as optional entries; however, they are important in the linked data domain (Goodwin *et al.*, 2008) and in the Web in general because they allow people to obtain and use the data that have just been discovered.

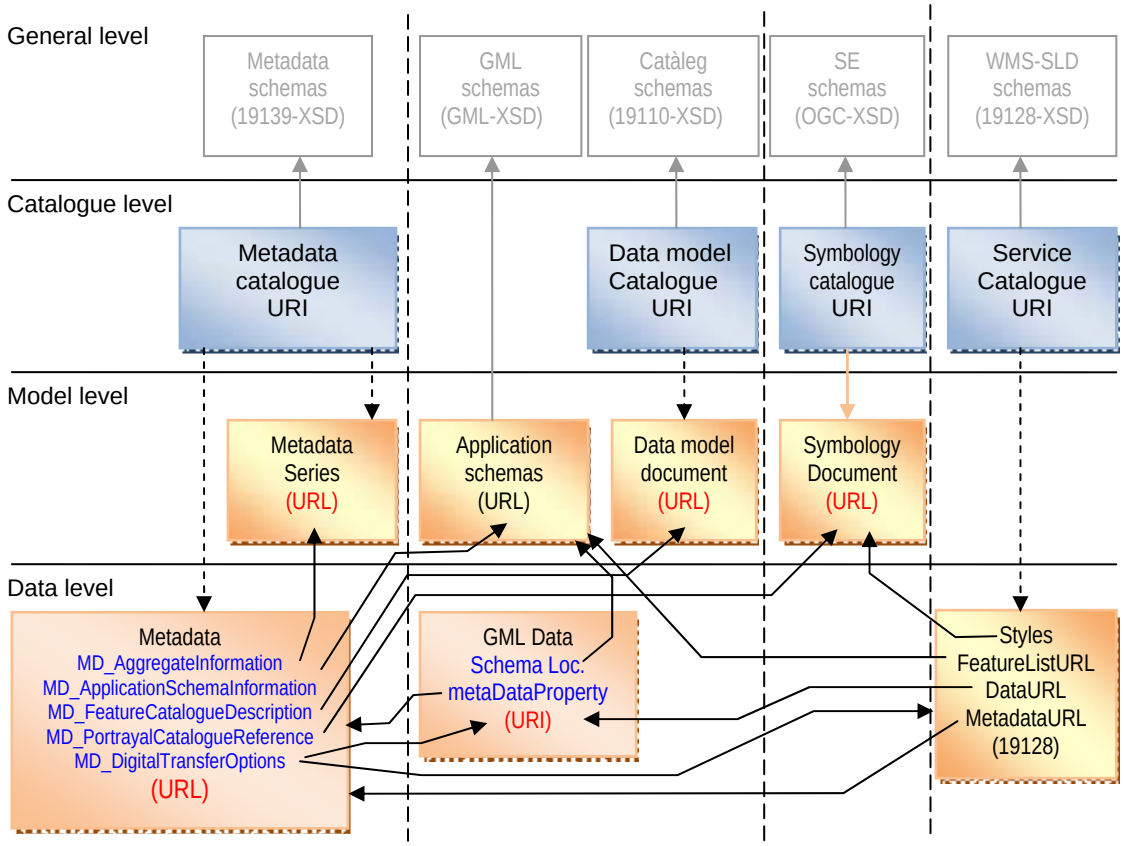


Figure 12: Relationships between different elements in data metadata, service metadata, and model standards, sometimes made by URL links, sometimes by identifiers (source: own preparation).

Manually edited metadata is painful to generate and error prone. New research shows that different methods of automatic metadata extraction (Han *et al.*, 2003) can generate homogeneous metadata records that are well synchronized with the data from GML (Olfat *et al.*, 2010) or image, digital terrain models and other vector formats (Manso *et al.*, 2009b), instead of been elaborated in a manual process. Additionally, there are only a few applications that correctly deal with dataset series and hierarchical metadata, which indicates that more research on this issue is necessary (Zabala *et al.*, 2005).

Semantic interoperability is still under research and is currently difficult to achieve, but it should be encouraged by promoting keyword dictionaries such us General Multilingual Environmental Thesaurus (GEMET) (Bernard *et al.*, 2005).

Improving metadata about services

There are at least three ways to encode the description of a service: ISO 19119 metadata description document, OGC Web Services Common “service metadata document” (ServiceMetadata) and Web Service Definition Language (WSDL). There is a need for harmonization and crosswalking among them. Also there is a clear need for relating service metadata with the data metadata they contain or provide access to.

Information characterizing the availability and reliability of the service can be obtained by testing the service from time to time, and results should be stored with the service metadata as reliability statistics. Also service usage metrics can be collected and added to metadata (Wang *et al.*, 2009).

Improving data models

Apart from the entity and attribute section 5 of the CSDGM, there are two generic ways of providing standardized data model descriptions: the ISO 19110 standard (specifies how to describe feature types and feature attributes of the dataset) and the ISO 19136 standard (that provides a set of rules for describing the feature types and feature attributes [GML properties]). The two alternative data model descriptions complement each other and should not be seen as different alternatives: More tools to develop and transform data models are necessary, even if these transformations require human intervention to complete the work (like the Snowflake software). More research and development should be carried out on data model transformations and cataloguing these models.

Some feature properties containing categorical information that can take a set of predefined values. Category dictionaries (thesauruses) should also be provided in some form; for example, they could be encoded in GML and linked to GML application schemas and instances.

Improving data download

Current SDI implementations are clearly focused on portrayal and map representation and do not provide enough information on the service side (or it is difficult to find) to allow applications and services to obtain and analyze the data. Spatial analytical tools require access to the data itself, but there are still few real data Web services (*e.g.*, WFS and WCS). In fact, Vandenbroucke (2008) demonstrated that in Europe, where many countries have complete metadata collections and map viewers, there is only a small percentage of data available for download (except for Austria and Norway). For practical reasons, second generation SDI have partially failed to achieve this aim and have involuntarily introduced confusion into the availability of the data due to, at least, three factors: first, the availability of metadata records seems to be enough; second, the Web services GIS industry has implemented WMS connectors (Bernard *et al.*, 2005) without providing download capabilities by any means; and third, when providers make the effort to offer their data, they prefer a standard way of doing this but when they deal with features they get confused on how to do that, because they have to confront GML but only a few software applications can read GML. There is a need for more tools making the use of GML easier. Meanwhile, there are a few formats that have become *de facto* standards (SHP, etc.) and there are very complete collections of data transformation tools (among these *de facto* standards) that work reasonably well; some of them are freely available, even in open source (*e.g.*, http://www.gdal.org/).

Improving data services and adding processing and sensor services

Geospatial Web services need to support a large number of concurrent requests, unleashing complex computations and requiring large-quantity data transmission (Tu

et al., 2004). Sometimes it is impossible to guarantee the quality of the service and the scalability of the design with a single computer, and a cluster of computers should be considered (Yang *et al.*, 2005). On the client side, clients have to be able to request data asynchronously in a multithreaded way, allowing users to continue interacting with the client instead of waiting for the previous request to get back. It is difficult to find the equilibrium between requesting only the part of the data the user really needs (Scholten *et al.*, 2006), or reducing the number of concurrent requests that the server will receive, and anticipating user data needs by transmitting larger pieces of information. However, the number of server requests can be reduced by using caching mechanisms in map and tile servers (Yang *et al.*, 2005), or by avoiding requesting single features and conveniently packing requests for feature collections (requesting features in a bounding box of several feature classes, etc) in a WFS server (Tu *et al.*, 2004). Chapter 6 evaluates the impact of concurrent requests in current map service implementations.

Some authors even state that current viewing and downloading Web services are the preliminary step rather than a complete solution, foreseeing geoprocessing as a necessary next step (Wright *et al.*, 2003). On the other hand, the emerging concept of cloud computing is seen as an opportunity to expand distributed geoprocessing. Primary SDI usages of geoprocessing are seen as coordinate transformation or format transformation services, but in fact WPS (Schut, 2007) could encapsulate interfaces for almost every geospatial analytical process.

Measure physical characteristics (*e.g.* temperature) and phenomena can be achieved by fixed sensors, or by sensors attached to vehicles or carried by people and connected to wired or wireless sensor networks (Craglia *et al.*, 2008). OGC has released a set of standards under the generic names of Sensor Web Enablement (SWE), and particularly OGC Sensor Observation Service (SOS) (Na *et al.*, 2007) should be adopted by SDI. It is necessary to harmonize sensor data with National Map Agencies (NMA) maps using data distribution, interpolation models, etc, in order to develop dynamic systems able to provide access both to raw sensor data for expert users, and to elaborated data.

Improving data portrayal and symbolization

To be able to meaningfully integrate spatial data from different sources, a uniform representation of spatial objects is required; however, often organizations are not willing to modify their presentation (or facilitate modification of the presentation by a third party) of spatial data to fit another purpose because they consider their symbolization as their branding. Therefore, most service implementations do not have this functionality (Vandenbroucke, 2008). Again there are 3 different ways to encode data symbolization: ISO 19117 or OGC Symbology Encoding (OGC-SE) and the default symbolization encoding in GML v. 3.

Web map browsers in SDI portals should provide several thematic presets to show the thematic diversity of the available data combinations. OGC Web Map Context (WMC) (Sonnet, 2005) is an excellent technological resource for defining presets because it defines a way of saving and retrieving WMS layer combinations (in fact, it is a resource that has been used for more that two decades in MiraMon MMM format or in ArcView

projects). The WMS and WMTS GetFeatureInfo or GetFeature operations should be used to show the alphanumeric data behind maps (Vanmeulebrouk *et al.*, 2009).

7.B.1.1.2. Evolving the hypermap concept: The World Wide Hypermap.

The main limitations of the old hypermap concept introduced in section 1.B.2.1.3 are: all queries must be previously defined and “prepared”, links are local identifiers, links have no purpose semantics, and users are limited to retrieving resources. There are four technologies presented here that applied together in a single system can overcome these limitations, such as, respectively: geospatial Web services and dynamically generated hyperlinks, the use of global geo-identifiers, add purpose to links and RESTful Web services to add the create, delete and modify resources capabilities. Once these limitations are overcome we can talk about a single World Wide Hypermap (WWH) with enhanced capabilities.

If you imagine a permanent collection of datasets (with all their different components) that represents an immutable world that considers all aspects imaginable, it is easy to generate a set of global identifiers in a centralized way, and then encode the logical links between them. But in the real world, datasets are generated progressively and frequently updated. A dataset is updated by creating new features or modifying some of the previous ones. Chapter 3 proposes a unified protocol to manage all these resources and a mechanism that generate and maintain a harmonized but decentralized global geospatial identifiers schema. The solution combines a new RESTful Web service with slightly modified classical GIS applications and generates a seamless WWH environment in which common GIS processes can be carried out directly in a classical GIS application or in a distributed environment on the Web. Local GIS instances can access data directly, while external users from the Internet can access data by requesting it from a RESTful application.

Proposed global identifiers are HTTP URI that are generated as soon as the resource is created. They also are maintained during the entire life cycle of these resources, even if elements are generated when the local GIS computer is offline. This guarantees the uniqueness of the global identifiers and avoids missing links to resources. To make this possible in a decentralized environment, the HTTP URI structure is separated into levels (or namespaces). The first level of the HTTP URI identifies the organization and is formed by the current URI of its main website followed by the fixed word "WWH" (using URI template language (Gregorio, 2008; Gregorio, 2012) this is expressed as http://{OrganizationServer}/WWH). The next level is the name assigned to the software instance used to generate the data (expressed as http://{OrganizationServer}/WWH/{SoftwareInstanceName}). The third level is the name of a geospatial set of entities (*e.g.*, a feature or a coverage collection) (expressed as: http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}. Next levels can identify particular features, coverages or even feature attributes of the geospatial entities and so on, as explained in Table 1 of chapter 3.

The uniqueness of the complete URI depends on 3 independent pieces of software that manage the first, second and further levels defined above (Figure 13). The uniqueness of the first level is provided by the HTTP naming authorities and

mechanisms (mainly the DNS registration). The uniqueness of the second level is provided by a centralized RESTful application inside the corporative domain. The uniqueness of the following levels is provided by each GIS software instance.

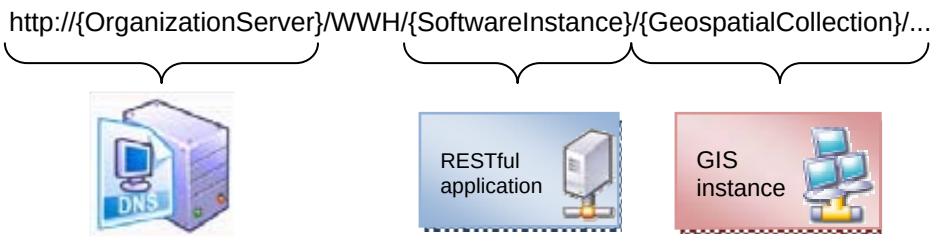


Figure 13: Entity responsible for maintaining the uniqueness of each of the 3 parts of a resource URI in the WWH (source: own preparation).

Each GIS software instance in the corporation can manage their own resources (more details in Figure 14 and Annex II). Resources managed are geospatial collections (feature collections, raster arrays or alphanumerical databases), geospatial entities (entities, cells or records), and complex attributes. Once identified, these elements can be hyperlinked to internal or external resources. In the WWH model, a hyperlink is a unidirectional relation between two resources of the WWH. Links use global identifiers, and a purpose for this relation can also be included.

Following this schema, GIS software instances are able to interact directly with their resources in their usual way (binary file accessing, geo-database querying, etc) and are also able to interact with remote features using the RESTful requests to the corresponding corporative application by applying one of the four HTTP methods and a global resource identifier. The four HTTP methods in the REST architecture (POST, GET, PUT and DELETE) are associated with the four common actions of any computer system (create, retrieve, update and delete) for each resource type.

There are five special resources that can be used by adding the words "_metadata", "_type", "_link", "_all" and "_history" to a given resource. The meaning is described in detail in chapter 3.

Apart form retrieving a single object, a very common operation with feature collections is to extract a subset of them according to some spatial and/or thematic criteria. This functionality can be achieved by the RESTful encoding with a POST request by adding the words “_SpatialSubset” and/or “_FilteredSubset” to a geospatial collection URI that includes en the body of the message a spatial object description encoded in, for example, GML, or a reference to it (*i.e.*, a direct reference or an OGC Web Feature Service (WFS) request that retrieves an object, typically a rectangular bounding box). The returned result is the URI of a new geospatial collection that can be retrieved later, like any other.

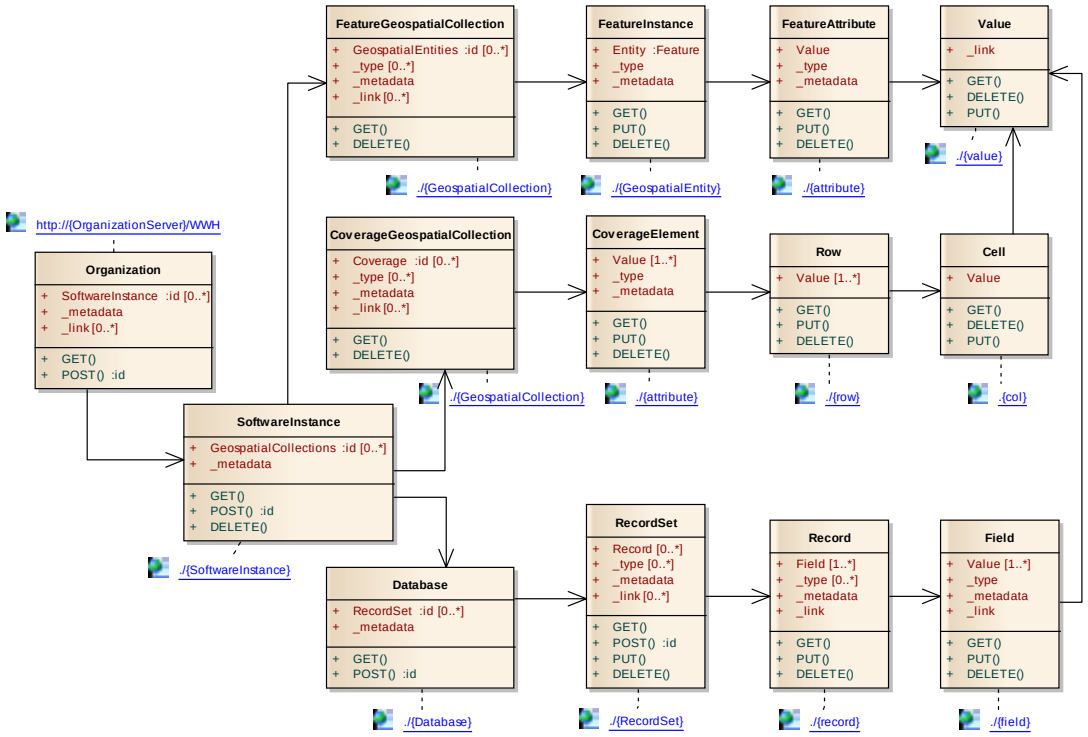


Figure 14: Summary of resources and their operations, relations and URI templates in the WWH (source: Figure 2, chapter 3).

Catalogues in the WWH

The proposed solution has several advantages over a classical approach based on RPC Web services. It eliminates the need for specific data access Web services as well as different identifiers for data, metadata and data access services. Actually, using a single identifier and a simple set of RESTful rules, users can indicate the data, obtain their metadata and gain access to the actual data. WWH also makes it completely unnecessary to catalogue data access services. OGC CSW metadata catalogues can be used to catalogue and search in metadata about data but it is also expected that common text searching Web tools and crawlers will also be useful for discovering data in the WWH.

Furthermore, catalogues can rely on the uniqueness of the original resource URI, so there is no need to generate their own local identifiers. The generation of different identifiers for the same resource is a problem that affects many current catalogues (Nebert, 2004). Another advantage is that metadata catalogues use the data URI as the base identifier so the responses of the catalogue will link to the actual data and users will know how to get them immediately.

Web Services in the WWH

Having an identifier for each geospatial resource from the moment the resource is created provides automatic access to any geospatial resource, and therefore there is no need for a specific feature or coverage service protocols. Since the data access service is implicit in the WWH, there is also no need to catalogue data access services

anymore. Therefore, this solution replaces current OGC WFS, WFS Transactional (WFS-T) and Web Coverage Service (WCS) protocols and functionalities, but it makes use of diverse OGC format encodings to exchange information. In fact, encodings are used to communicate resource representations and GML, KML, GeoRSS (Wick *et al.*, 2007) and Filter Encoding language are excellent recommended encodings for geospatial representations in the WWH. However, map services, including the tiled ones, can be offered by each geospatial collection directly using the WMS (de la Beaujardiere, 2004) or WMTS (Masó *et al.*, 2010a) protocols, considering the geospatial collection URI as a root server of the OGC service and adding the WMS Key-Value Pair (KVP) specified parameters (KVP or REST for the WMTS).

Finally, the RESTful application can be extended to provide remote execution support for processes with internal or external data. Processes can be executed directly by the GIS application or can be executed by posting an OGC Web Processing Service (WPS) execute document (Schut, 2007) to the http://{OrganizationServer}/WWH/_SpatialOperation URI.

A comprehensive list of resources and operations defined around the WWH idea can be found in the Annex II of this PhD thesis.

7.B.1.2. Use Cases

In this section, two application cases are presented as deployments that apply the results of the previous conceptual results. The first is described at the end of chapter 2 and the second is described at the end of chapter 3

7.B.1.2.1. Web *2.0 and metadata user feedback: IDECTalk*

Current standard-based approaches to metadata capture require considerable human input, and are difficult to keep up to date. They primarily represent the perspective of the data producer (on quality and utility of the data). Since quality is currently defined as fitness-for-purpose, user feedback is essential for expressing the users' measures and opinions about the dataset.

Some of the current limitations of metadata could be overcome with more participative methods of user qualification and feedback, as is already a common practice in commercial Web 2.0 services. In this application case we put into practice some aspects previously discussed in the section Adding Mass Marked, VGI and Web 2.0 of chapter 2.

A pilot project which allows VGI about metadata is presented demonstrating that some of these problems (that also affect the Catalan SDI) can be overcome with a search portal connected to the SDI metadata catalogue. Indeed, this portal allows metadata about a particular topic to be requested to identify a particular dataset. The user can read the catalogued metadata from the producer and also the previous user comments. Moreover, they can enter new comments or update on their own. All of this is stored and becomes immediately available to other users.

The environment is a mash-up that relies on the Catalan SDI catalogue and mass market mapping tools (Figure 15). It is a CGI application connected to a database that stores user comments and that resides in a Web server. We use the Universal and Unique Identifiers (UUID) that the Catalan SDI assigns to each metadata record to link both official metadata records and user comments. The dataset bounding box extracted from the metadata record is also shown using the Google Maps API, as another mash-up (Chow, 2008).

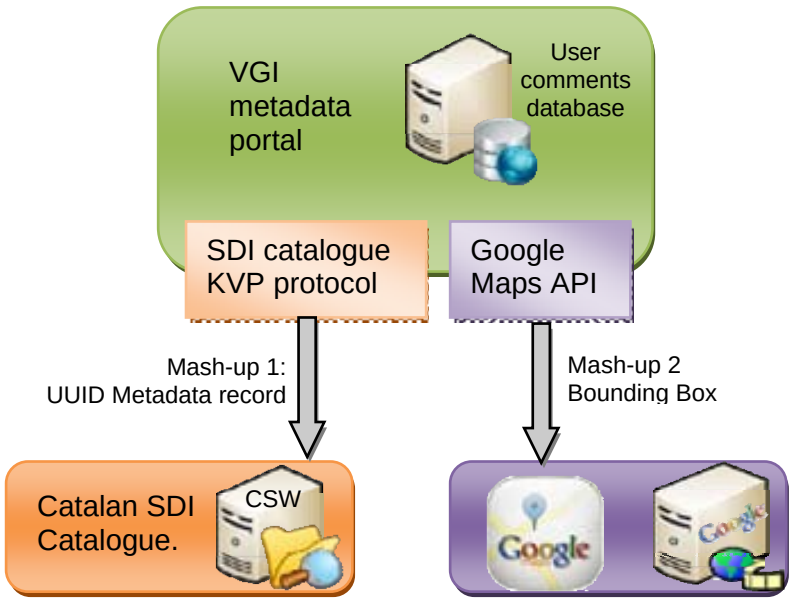


Figure 15: IDECTalk architecture. Mash-up 1 connects to the Catalan SDI using a UUID and mash-up 2 connects to Google Maps using the Bounding Box (Figure based on Figure 5 in chapter 2).

The experimental environment deliberately uses a style that is similar to Web search engines and *blog* applications so the user feels immediately familiar with the system. This experimental environment was developed in less than a week and no direct communication with the SDI support centre was required. This demonstrates that setting up Web 2.0 applications that consider users' comments connected to an SDI metadata catalogue can be easily deployed by any third party, who immediately becomes part of the SDI infrastructure.

Clearly, there are some issues that need to be addressed more in depth, such as the real motivations of people providing volunteer information and the process needed to ensure the quality of the information provided (Craglia *et al.*, 2008); however, we believe that the experiment is an interesting starting point. Future lines of extension of this platform are the introduction of user tagging capabilities (Kalantari *et al.*, 2010) as well as to collect data about the user behaviour and adding this information to the system.

7.B.1.2.2. MiraMon-REST

The implementation of the WWH requires some extra software that make requests on top of the HTTP protocol. As a reference implementation we discuss how it can be implemented in an extended version of the MiraMon software. This results in a file based implementation, but it is possible to implement it using other data systems, such as a geodatabase.

The extension of the MiraMon software can be explained as a set of modifications on the MiraMon GIS Professional software and the extension of the MiraMon Map Server (MMS) into a RESTful service (Figure 16). The two applications share a common core of functionalities but also have some particularities. In the common core, both applications are able to perform the maintenance of the internal data files directly and both can act as RESTful clients using HTTP. They are able to make requests like the ones presented on section 7.B.1.1.2 and included in chapter 3 (listed in Table 1 and 2, and explained in section 3).

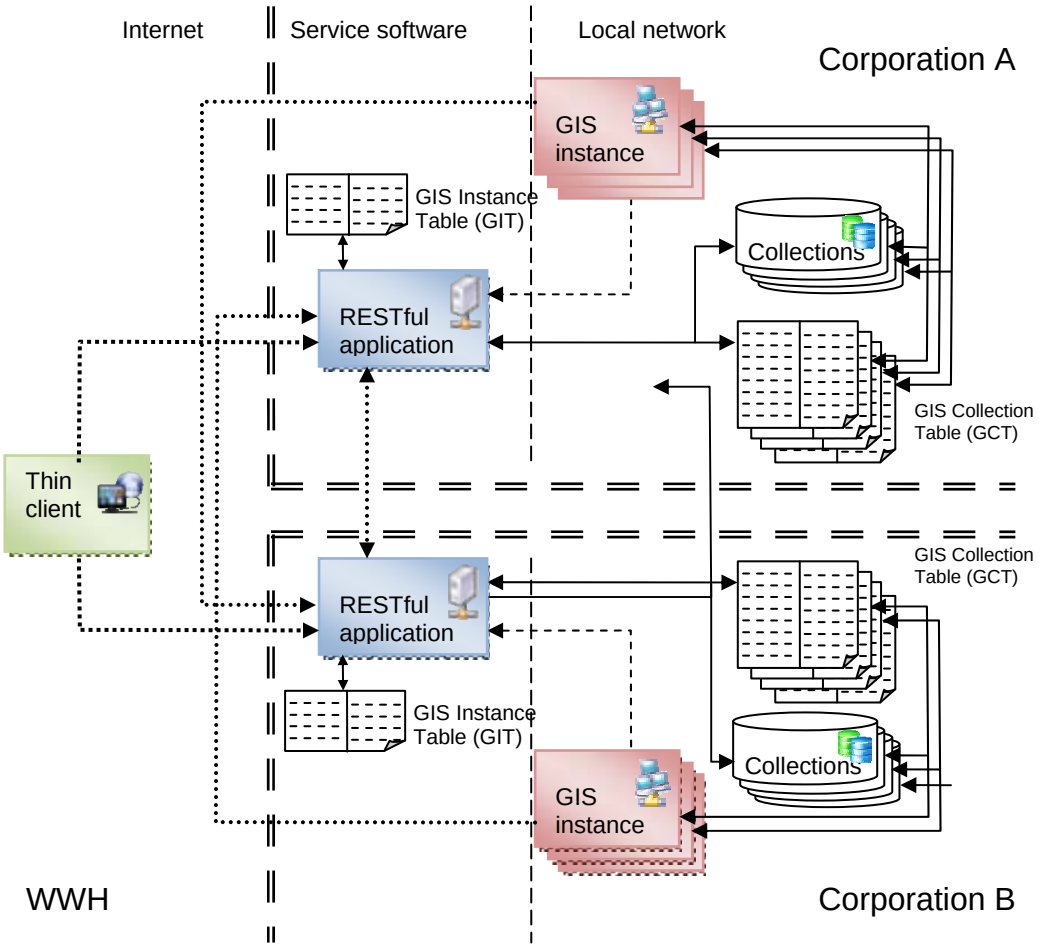


Figure 16: Software components and connections between two corporate nodes and an external thin client in the WWH (source: Figure 3, chapter 3).

Modifications in the GIS internal component

The main functionalities of the MiraMon software do not need to be altered, but small additions need to been made. The internal file data model have to be modified (keeping backwards compatibility) to allow the collection identifier (recorded as a metadata field) to be stored, and hyperlinks to geospatial collections (recorded as metadata fields) and to geospatial entities and attributes (stored in the attributes database). Support for references to external geospatial entities in geospatial collections need to be added. The ability to function as a RESTful client (and make a request to any other RESTful service in the WWH) need to be added.

The MiraMon desktop software can use the methodology previously described to access the URI and get the list of geospatial collections available. The geospatial collections are then transferred and shown to the user. Another important modification is the addition of the Geospatial Collection Table (GCT) that keeps track of the identifiers of all the geospatial collections (datasets) maintained by a particular GIS instance and the access method required for managing the data (*e.g.*, a file path or the geodatabase access parameters).

Corporative RESTful server

This application is an extension of the MMS HTTP CGI capable of responding to RESTful requests. It needs to recognize the four HTTP methods and translate them into actions that have to be performed on the identified resources. This way it is able to recognize a global resource identifier, to find the GIS instance that keeps the data, discover where the actual resource is stored and to perform the requested operation directly on the data. To make this possible, it keeps a list of all GIS instances in the corporation (in a GIS Instance Table, GIT). When a new GIS software instance is set up on a particular internal computer it also sets up its own GCT. In addition, it creates a resource identifier with a POST operation to the http://{OrganizationServer}/WWH URI sending information about how to access the GCT. This makes the RESTful service to register the new GIS instance in the GIT.

When a corporate RESTful application receives a request for internal data it receives the global identifier. This identifier can be separated into 3 levels. The first level is used to identify the request as a URI to an internal resource and to accept the request. The second level allows it to identify the software instance and read the GIT to determine the way to access the GIS instance table (GCT). The third level provides information about the geospatial collection requested; the GCT translates it to information on how to access the real data that can be accessed directly. The RESTful implementation can also act as a GIS software so that it has a reserved software instance identifier, http://{OrganizationServer}/WWH/_external, and also manages its own GCT for collections that are created from outside without any internal GIS software instance intervention, such as requests for spatial filters, thematic filters or process results, or by simply sending new data with a POST operation.

7.B.2. Specific aspects of the Internet map browsing

The research done on Internet map browsers has given a set of results that are summarized in this section and where mainly extracted from chapters 4, 5 and 6 of this PhD thesis.

7.B.2.1. Conceptual and architectural aspects

This section summarizes the results more conceptual and about architecture that help to improve interoperability and performance for map services. To achieve it, a new standard is proposed and the results of the analysis to generate a map services based on JPEG2000 standard are summarized.

7.B.2.1.1. Web *Map Tile Service (WMTS)*

The WMTS (see chapter 4) provides a standard based solution to serve digital maps using predefined image tiles. The service publishes the tile schema adopted and makes it available (named TileMatrixSet, that is composed by a list of available scales and a regular tile pattern for each scale [Figure 17b]) in the ServiceMetadata document, that is common in all OGC Web services. This declaration defines the tiles available in each layer (*i.e.,* each type of content), in each graphical representation style, in each format, in each coordinate reference system, at each scale, and over each geographic fragment of the total covered area. A part from being able to return tiles, the service also provides the optional FeatureInfo capability to return descriptions of a point over a tile at specific locations.

This standard specifies several different concrete exchange mechanisms among clients and servers in two different architectural styles. The standard defines the GetCapabilities, GetTile and optional GetFeatureInfo operations for procedure oriented architectural style based approaches using several different message encodings, including messages encoded using Key-Value Pairs (KVP), XML messages, or XML messages embedded in SOAP envelopes. The standard also defines the request mechanisms to enable a resource oriented architectural style based on Web based URL endpoints, allowing clients to simply request the ServiceMetadata, Tile, and FeatureInfo resources as documents. Finally, it makes some recommendations to improve interoperability between implementations and one of these is the adoption of well known scale sets (WKSS) that are a common set of scales in the WMTS standard or by recognised organizations and that are used by many different servers, making the combination of their layers in a single client easier (Figure 17a).

As a result of this standard release, several implementations have been made available and several scientific papers already reference the relatively new normative document in chapter 4.

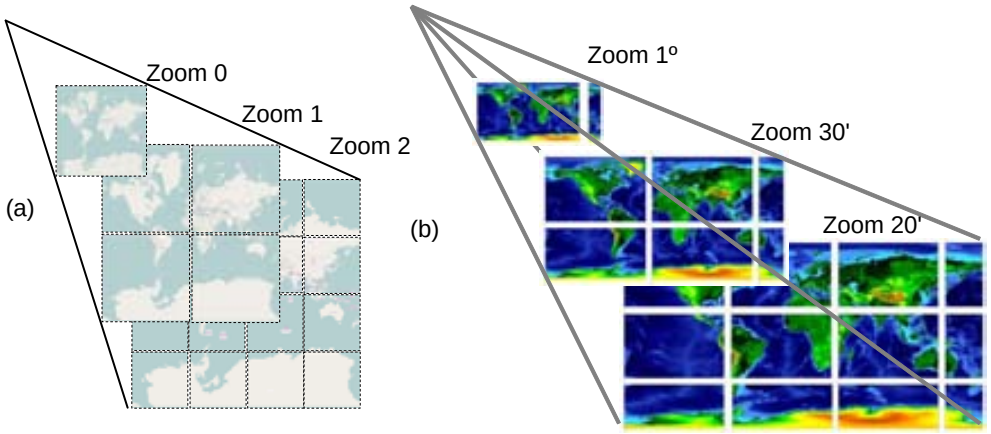


Figure 17: Two different TileMatrixSets composed by TileMatrix's that uses different scales sets (3 visible in the figure). (a) is using the GoogleMapsCompatibleWKSS (tiles come from OpenStreetMap); each TileMatrix is divided in squared regular tiles. (b) is not following a WKSS and uses rectangular tiles (source: own preparation).

7.B.2.1.2. Using JPEG2000 in map services

To understand this discussion you have to keep in mind that a JPEG2000 can define a tile structure only for the most detailed resolution but the compression is done in the transformed wavelet space. In order to optimize a JPEG2000 file to be served as WMTS, we are going to tile it in the same pattern that the WMTS tiles that we are going to expose for the base resolution level. A JPEG2000 based WMTS server has to define the next scale (with scale denominator double of the previous one) with the same origin (TopLeftCorner) and the same tile width and height in pixels (that results in tiles that covers a region double [4 times in area] of the previous scale level). Following this schema, the extraction of a tile WMTS in the base resolution level will involve one JPEG2000 tile and data from all internal JPEG2000 resolution levels in this tile. The extraction of a tile of the second WMTS scale level will require reading data from 4 JPEG2000 tiles and accessing codeblocks from all JPEG2000 resolution levels except the most detailed one. The extraction of a tile of the third scale level will require reading data from 16 JPEG2000 tiles and accessing data from all JPEG2000 codeblocks of all resolution levels except the 2 more detailed, and so on (Figure 18).

In some cases, where many WMTS resolution levels are needed and huge images are involved, coarser resolution levels can be difficult to obtain from a complete JPEG2000 file because the number of JPEG2000 tiles involved. For that reason, we recommend to generate some (2, 3, …) independent images of chained resolutions and combine them together on a single WMTS layer. JPEG2000 file that will be used in a WMTS will perform much better if it is prepared (compressed) with the small tiles typically used in a WMTS service (i.e., 256x265 or 512x512).

This structure of levels is exposed in the layer description of the WMTS service metadata document as a base scale level and the next resolution levels will share the same origin and sizes up to the number of resolution levels in the JPEG2000 files (the

client is not aware of having internally different file inputs). Then a WMTS server that receives a WMTS request will have to translate it to JPEG2000 decompression routine commands (Figure 19).

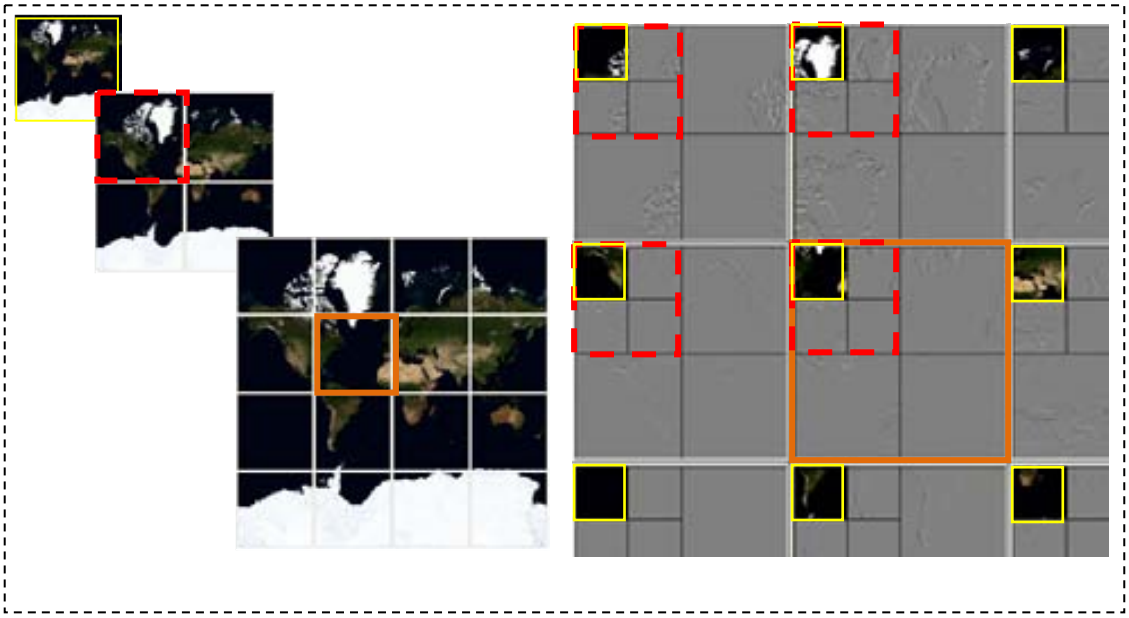


Figure 18: Tiles at 3 scale levels and correspondence between *codeblocks* in a 3 resolution levels of the wavelet transform: orange represents equivalences in the base resolution level, red equivalences in the second resolution level and yellow equivalences in the third resolution level (incomplete for readability reasons, source: own preparation).

Comparing WMTS and JPIP services.

Generally speaking, JPIP protocol could be considered an alternative to WMTS since they share the common interest of providing a protocol for disseminating images over the net in a fast way. There are 2 main differences between WMTS and JPIP: The first relies on the fact that WMTS typically uses common simple formats (PNG, JPEG, etc) and can eventually use JPEG2000, but JPIP uses JPEG2000 only. For that reason, WMTS can be used in a Web browser client without any extra software and this makes it ideal for combining it with other Web applications; on the contrary, JPIP requires a thicker and more specialized client. The second difference is that JPIP protocol allows maintaining codeblock cache on client and server, so that codeblocks received in previous transmissions do not have to be transmitted again. This property saves time and bandwidth when requesting a different zoom level from a region of the image that was previously requested with another resolution but requires the server knowing about the state of each client, either keeping a session on the server or by transmitting the client state to the server in each request. Current specification of WMTS cannot include that and, in the same circumstances, WMTS implementations transfer some redundant information. Nevertheless, WMTS is designed as a completely stateless protocol, so that it can take advantage of the Web caching mechanism that will eventually benefit all the WMTS concurrent users of a particular server, making WMTS more appropriate that JPIP for very popular Web map services. Also, WMTS has a RESTful interface that allows deploying a WMTS service in a standard Web server

software (*i.e.*, Apache, IIS, etc) as a set of static prerendered files, with no extra software needed (making the deployment process compatible with all current Internet service providers).

Future implementations of the WMTS standard could be extended in a way that eliminates some redundant transmissions if a WMTS client has previously received a tile image for the same region of a coarser resolution without breaking compatibility with the current WMTS specification. A possible technique could consist in transmitting the coarsest resolution tile and then, when a more detailed resolution is requested, only the differential information needed to enrich the previous tile to this new level of detail is sent. This can be achieved by using a Laplacian pyramid described by Burt *et al.* (1983). Instead of storing complete true images for each level of the pyramid (like the Gaussian pyramid does), in a Laplacian pyramid, the information stored for each particular level is constructed as the difference between the Gaussian image of this level and the Gaussian image of the next coarser resolution. Having a coarser image and the Laplacian information for the next level it is possible to reconstruct the actual image for the next level. However, a hypothetical WMTS service that supplies Laplacian resolution levels will require a thicker and specialized client that stores previous coarser resolutions and combine information from the next levels to show the actual image. This approach will have a set of requirements similar to a JPIP client and will not have the extra JPIP protocol advantages in quality scalability, resolution scalability, and flexible spatial random access (Patel *et al.*, 2007), but it will still remain a stateless solution that will properly use internet caching mechanisms and have better concurrency scalability.

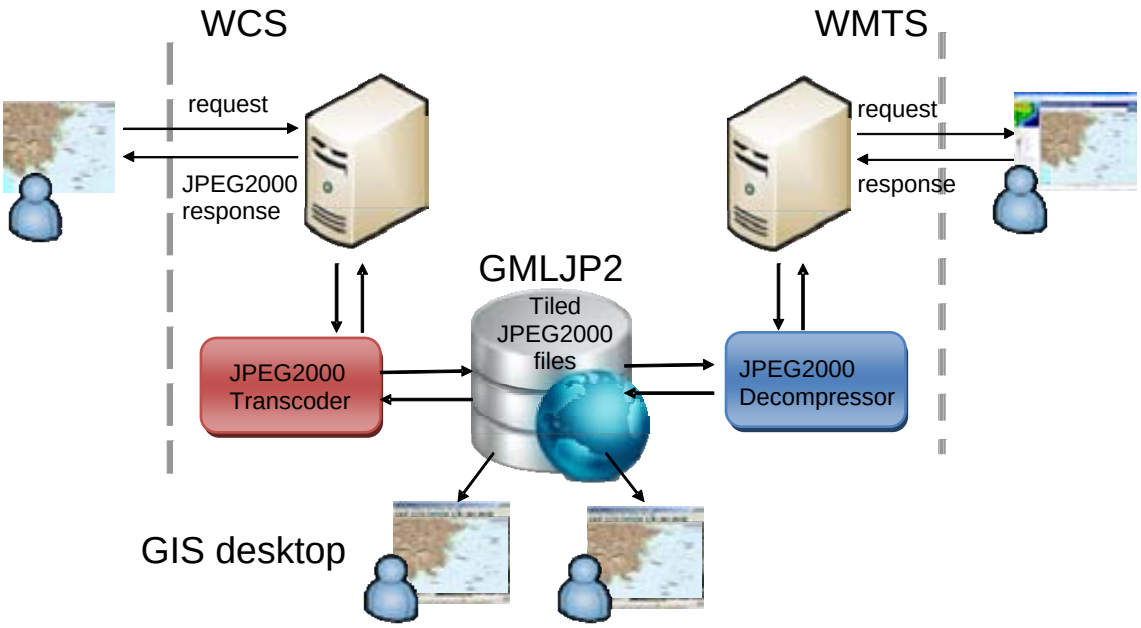


Figure 19: Single tiled JPEG2000 images repository (with metadata encoded in GMLJP2) for either internal GIS desktop, WCS server and WMTS server. (source: Figure 9, chapter 5)

Finally, there is another reason to use JPEG2000 format: A corporation can have a data storage with a collection of JPEG2000 files that can be used as internal repository for WMTS, WCS or SOS data services and, at the same time, directly used in the corporative RS or GIS application completely avoiding duplicity of data formats and providing a highly efficient compressed storage format (Figure 19)

7.B.2.2. Use cases

This section summarizes the results of a performance detailed analysis of map services (WMS and some based on tiles servers, including the WMTS explained in chapter 4). The results of this section justify the standard WMTS approach and expose how different strategies for different manufacturers affect the performance. Among the manufacturers we highlight the product Express Server (by Lizardtech) that uses a wavelet format very similar the one used in JPEG2000 analyzed in section 7.B.2.1.2 and chapter 5.

7.B.2.2.1. Maps server performance analysis

This section exposes the evaluation done upon seven of the most popular server solutions (GeoServer, MapServer, MiraMon Map Server, Express Server, ArcGIS Server, TileCache and GeoWebCache), for map service standards (WMS, WMTS, WMS-C, TMS), and determines response time, user functionalities and usability. Particular attention receives the different tiled approaches using sequential, concurrent and semi-concurrent implementations.

Evaluation of concurrent requests to a single server

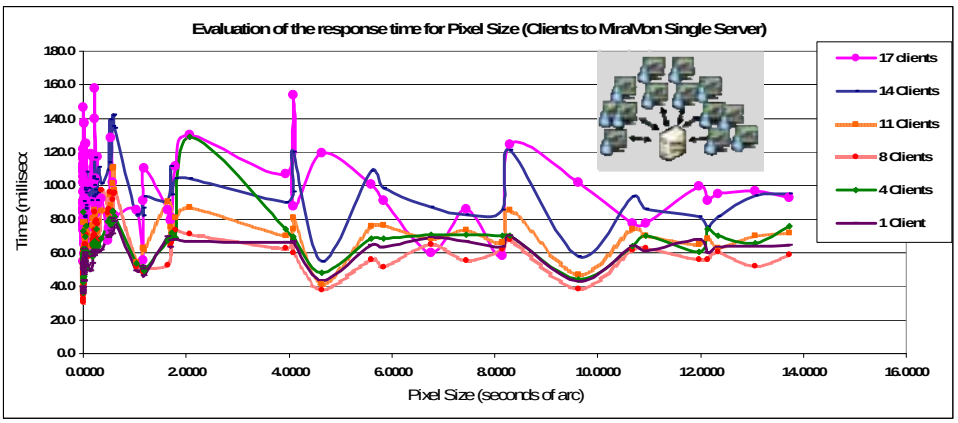
One of the main factors affecting Web server performance is request concurrency. Chapter 6 measures both the influence of the pixel size and the image size in the time response in WMS requests. More than one hundred different requests were done and these were made from up to 6 simultaneous clients. One immediate result in all measured circumstances is the increase of the time response when increasing the number of simultaneous clients requesting a single server. The fastest server is Express Server, probably due to the nature of the wavelet compressed format (MrSID[37] or JPEG2000) that is internally organized in a pyramid of zoom levels. ArcGIS Server is the server using original GeoTIFF images that has the better results. As an intermediate performace we find MiraMon Server which requires a prerendering process. Since MapServer is programmed in C language, a better performance was expected in front of the Java based GeoServer. These expectations were confirmed with a single client requests, but GeoServer performed faster than Mapserver for concurrent requests. There are also small differences in time responses depending on the output format requested, being JPEG the fastest format in MapServer, Express Server, ArcGIS Server and MiraMon Map Server, but PNG in GeoServer.

[37] Pronounced Mister Sid

Evaluation of a cluster of servers

In the current hardware state of maturity it is not possible to dramatically increase performance by getting a faster machine even if you are willing to pay more. Current computer technology seems to have reached a speed limit in CPU processing time and disk speed access. To overcome the performance degradation observed in concurrent requests a possible alternative solution is to set up a cluster of servers that can act as a virtual single server that deals with requests in parallel. We carried out tests comparing a WMS single server and a WMS in a computer cluster server in Network Load Balance[38] (NLB) configuration, in which 6 computers are able to respond at same time to different clients as if these were like a faster single server. These tests have been made on up to 17 simultaneous clients to evaluate the response time for the MiraMon Map Server case.

(a)



(b)

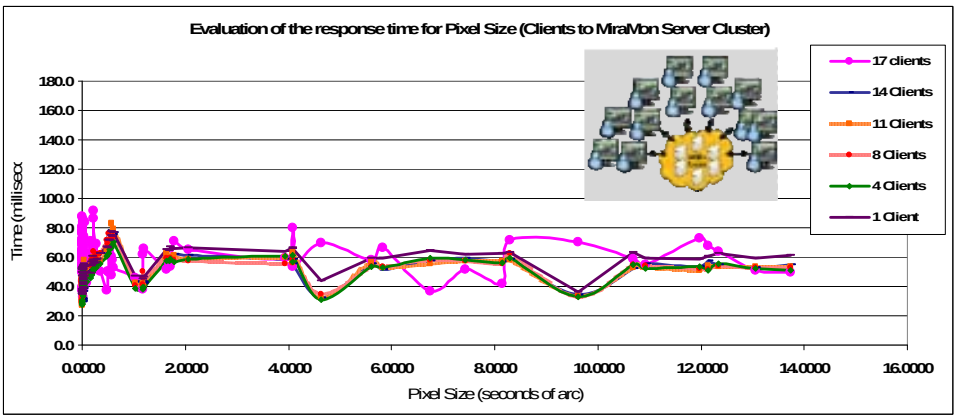


Figure 20: Time response for different concurrent requests for up to 17 clients to a MiraMon WMS server: (a) single server configuration (source: Figure 4, chapter 6), (b) cluster of 6 servers configuration (source: Figure 5, chapter 6).

Time response measurements comparing a single server and a 6 computer cluster stressed with multiple client requests reveal that NLB server response time is much more stable and almost equivalent to the single client stress case even for 17 simultaneous requests (as shown on Figure 20). We expect some degradation if we

[38] NLB is one of the simpler computer cluster approaches where several computers share the same web address and the same content. When a request comes from a web client, it is assigned to a single computer that will dispatch it. Next request will be assigned to a different computer in the cluster.

increase the number of simultaneous request even more, but fortunately NLB cluster performance can be improved again by aggregating more servers to the cluster (up to 64 in Windows 2003). If we suppose that the performance is linear, this means that this configuration can be scaled at least to serve up to ~200 simultaneous requests without performance degradation. Please note that the time response for these requests is always lower that 0.1 second, so this configuration is equivalent to 2000 request per second.

Tiled request and responses

In the previous chapter we have been using a common WMS interaction where a WMS client requests the whole image needed to cover the client viewport in a single piece (as previously explained in Figure 6). Recently, some WMS clients (like OpenLayers) are able to tile the space in a regular matrix of small pieces. By doing so, several tiles to cover the whole viewport are needed but the client can recycle some tiles when the user moves the view laterally and also can take advantage of the cache mechanisms. However, this strategy can have its drawbacks if the caching mechanism cannot help and the server has not been prepared to manage this situation because a view is composed by several tiles that are concurrently requested to the server, and, as we have discussed previously, the response time increases in simultaneous requests, although users might not have this perception because some tiles get the client sooner and are shown immediately. The results we are exposing here clarify the effects of this approach on a classical WMS server and quantify the difference between fast full image delivery and tiled images in a way agnostic of the user perception. Improvements to these situations by applying tile strategies directly to the server, such as OSGeo WMS-C and OGC WMTS that have been also studied.

We have simulated tiled clients (tiles of 256x256 pixels) that make requests to common WMS (which have no particular strategy to deal with tiles) in the three following configurations: Tiled WMS in unlimited concurrent requests, Tiled WMS in semi-concurrent requests in packages of four tiles at the same time, and tiled WMS in sequential requests. We have also compared these three configurations with a regular WMS full viewport image. Optimum results depends on the product analyzed but full WMS requests and semi-concurrent requests are the fastest configurations.

Also products that are optimized for tile responses were tested (they provide a way of prerendering tiles or saving tiles that are generated on the fly for further use). The main drawback is the generation time, but this can be partially overcome by metatiling strategies: Instead of generating each tile individually, a metatile is generated creating a single larger map image that is divisible into a number of tiles. For all servers analyzed a 512 x 512 image requires more time but much less time the sum of the time required for the four 256 x 256 images.

8. CONCLUSIONS

8.A. Conclusions (versió en català)

Aquesta tesi estudia aspectes genèrics derivats de l'aplicació dels estàndards geospacials, per centrar-se més endavant en els serveis de mapes. També presenta algunes solucions pràctiques per millorar els estàndards i les seves implementacions.

El fenomen de les Infraestructures de Dades Espacials (IDE) ha anat creixent contínuament durant les darreres dues dècades. En aquest procés, i en especial en la implementació de la seva segona generació, ha estat crucial l'adopció d'un model descentralitzat impulsat per estàndards internacionals i tecnologies d'Internet. No obstant això, hi ha diverses qüestions en el desenvolupament de les IDE que necessiten ser resoltes, com s'ha demostrat en un cas pràctic d'ús relativament senzill, en el qual s'intenta fer servir la IDE de Catalunya (IDEC) per estudiar l'accessibilitat als centres de salut del país, que demostra que les IDE fracassen, tant en proporcionar informació sobre les dades disponibles a l'administració catalana, com en proporcionar eines distribuïdes d'anàlisi.

El rendiment de les IDE pot millorar sense una reconceptualització dels seus principis, sinó més aviat reforçant o reorientant alguns aspectes de la generació actual de les pròpies IDE. El capítol 2 proporciona una llista d'aspectes que influeixen en el rendiment i la facilitat d'ús i que són millorables; la llista es presenta de manera que es manté l'equilibri entre les eines disponibles en els portals i els serveis web que hi ha darrera d'ells. La recol·lecció de metadades i els processos de cerca sobre elles s'han de millorar, així com també s'ha de clarificar les pràctiques sobre les de metadades de servei perquè tant les dades com les eines que funcionen en un entorn distribuït es puguin trobar fàcilment. La disponibilitat de les dades reals amb la seva simbolització per defecte ha de ser combinada amb els serveis web estàndard en un entorn integrat que, a més, pugui ser enriquit amb informació geogràfica de voluntariat (*Volunteered Geographic Information* [VGI]). Per demostrar que la VGI pot ajudar fàcilment en el desenvolupament de les IDE, es descriu un exemple d'un lloc web que barreja protocols estàndard clàssics amb els moderns *mash-up* de la web, la qual cosa permet que voluntaris complementin els catàlegs de metadades amb la perspectiva de l'usuari i, per tant, es cobreixi les llacunes en la informació proporcionada pel proveïdor.

Les IDE, i el SIG distribuït, també es poden millorar mitjançant l'aplicació de la reformulada arquitectura RESTful, que amplia el concepte hipermapa, definit fa dues dècades, per tal de superar les seves limitacions originals. L'estil de l'arquitectura orientada al recurs oculta la complexitat dels serveis web i unifica les dades, les metadades i els serveis d'accés a les dades, a partir de reutilitzar principis simples que s'han anat fent servir a la web des dels seus inicis. Així, l'adopció d'identificadors de recursos universals (URI) en un entorn distribuït permet construir una xarxa de dinàmica d'hipervincles a recursos geospacials que defineixen un hipermapa únic i global. Les IDE actuals han anat posat en marxa serveis de d'estil RPC-SOA basats en estàndards internacionals OGC i ISO des de fa més de 10 anys, els serveis web de mercat de masses i de les comunitats VGI utilitzen principalment els serveis RESTful. El concepte WWH torna a connectar aquests dos mons i facilita la introducció a ells, proporcionant un entorn harmonitzat i adoptant la web semàntica. El GEOSS i

l'INSPIRE són només dos exemples de grans sistemes de dades geospacials compostos per un conjunt de servidors de dades i sistemes distribuïts que utilitzen estàndards d'accés de dades i protocols heterogenis per a difondre dades científiques i comercials. Mitjançant l'adopció d'un gran nombre estàndards, ambdós sistemes volen millorar la interoperabilitat, però encara s'enfronten a algunes dificultats. L'INSPIRE intenta superar-les mitjançant la definició de guies i normatives de desenvolupament, així com limitant el conjunt d'estàndards acceptats a un perfil reduït per al descobriment en un marc comú d'accés a dades. Malgrat tots els esforços en aquesta direcció, cap d'aquests sistemes té un mecanisme integrat per l'actualització i el manteniment de noves dades. L'hipermapa mundial d'Internet (WWH) proporciona una interfície unificada per al descobriment de dades, així com l'accés a dades i serveis, i relaciona entre si els coneixements, millorant l'accés a les dades geospacials i els resultats d'estudis científics. Funcionalitats típiques, com el filtrat espacial o temàtic de col·leccions, la generació de mapes, l'execució de processos, etc, també s'inclouen en l'enfocament RESTful, de manera que els serveis d'accés a dades (per exemple, WFS i WCS) esdevenen innecessaris i la integració d'altres estàndards de serveis web com ara el WMS, el WMTS i el WPS queda contemplada. Es presenta una implementació de referència que demostra que només unes extensions i modificacions menors al programari SIG són necessàries perquè aquest pugui ser incorporat al WWH. Aquest exemple, es complementa amb una aplicació CGI RESTful corporativa que actua com un intermediari, connectant les aplicacions i els arxius al WWH.

En particular, en l'àmbit dels serveis de mapes, l'estàndard WMTS, presentat en aquesta tesi doctoral, constitueix un servei de mapes tessel·lats dissenyat tenint en compte el rendiment i l'escalabilitat des del primer moment, el que permet que les implementacions dels servidors WMTS puguin retornar tessel·les molt més ràpidament. Per aconseguir-ho, el WMTS converteix l'espai continu en un de discret, definit per un nombre limitat de nivells de zoom preestablerts (escales) i un espai regular tessel·lat, tot assignant un índex triple per cada tessel·la. Una bona manera d'aconseguir un bon rendiment és emmagatzemant localment tessel·les prerasteritzades que no requereixin cap tipus de processament d'imatge o geoprocessament addicional. Això es pot aconseguir, prerasteritzant totes les tessel·les en un procés de preparació anterior, o generarant-les, sobre la marxa, i guardant-les en la memòria *caché*. La interfície WMTS estableix que un client pot obtenir tres tipus de recursos, ja sigui en resposta a una sol·licitud en l'estil d'arquitectura orientat a recurs o en resposta a una operació en l'estil d'arquitectura orientat a serveis. Aquests recursos i operacions són: un recurs ServiceMetadata (resposta GetCapabilities) que descriu les capacitats i la informació continguda en un servidor específic: un recurs de tessel·la (resposta GetTile) que mostra un fragment d'un mapa que representa d'una fragment d'un capa, i un recurs FeatureInfo (resposta GetFeatureInfo) que proporciona informació sobre les entitats situades en un píxel particular d'una tessel·la. El WMTS té moltes similituds amb altres serveis web de l'OGC (OGC Web Services [OWS]), com ara el WMS, el WFS, i el WCS, però, ofereix, apart de la clàssica sintaxi KVP, els protocols SOAP i RESTful.

Una possible implementació del WMTS es pot beneficar també de l'ús d'imatges JPEG2000 com a format d'emmagatzematge intern, però cal tenir en compte que encara que les imatges JPEG2000 tinguin una estructura de *codeblocks* que permeti

extreure fàcilment d'una regió de la imatge a qualsevol resolució; a fi de tenir una resposta el més ràpida possible, el JPEG2000 necessita fer servir un model intern de tessel·les JPEG2000. Els conjunts d'escales i matrius de tessel·les WMTS es poden restringir mitjançant un petit llistat de regles que permetin que el servidor respongui més ràpid amb formats comuns o amb petits fitxers JPEG2000, i com a avantatge addicional, una corporació pot utilitzar un magatzem de dades únic amb un recull d'arxius JPEG2000 que poden ser utilitzats com a dipòsit intern per al servei WMTS, alhora que també serveis com el WCS, el SOS i el JPIP en poden fer ús, evitant la duplicitat de formats de dades i proporcionant un fitxer comprimit d'emmagatzematge d'alta eficiència.

Aquesta tesi també analitza la resposta de diverses implementacions de servidors de mapes entre els més comunament emprats. Com a primer resultat, tots els servidors analitzats són més lents quan el nombre de clients simultanis s'incrementa. Per resoldre aquesta situació, és una molt bona opció instal·lar un *cluster* de servidors (un grup d'equips que són capaços de respondre a la vegada a diferents clients, assignant a cada client un equip diferent del grup). Els resultats quantifiquen com de pitjor és el rendiment dels servidors WMS que no estan optimitzats per a la resposta de tessel·les si els clients els sol·liciten tessel·les igualment. El MapServer i el GeoServer amb la configuració mínima no requereixen cap procés de preparació de dades, però el seu rendiment és pitjor que altres serveis que requereixen de mètodes d'indexació com el Servidor de Mapes del MiraMon o l'Express Server. El MapServer (basat en el codi C++) es comporta millor que el GeoServer (basat en el codi de Java) en sol·licituds no concurrents, però el GeoServer és sorprenentment més ràpid sota peticions simultànies. El servidor WMS més ràpid que hem provat és l'Express Server, que treballa internament amb imatges MsSID o JPEG2000 comprimides que tenen una grandària de només un 5% de la mida original sense comprimir mentre que el servidor de tessel·les més ràpid en els tres casos avaluats (peticions concurrents, semi-concurrents i seqüencials) és el GeoWebCache construït sobre el GeoServer.

Línies de futur: La recerca exposada en aquesta tesi desenvolupa només alguns dels temes teòrics i aplicats relacionats amb el SIG a Internet i els estàndards per a serveis geospacials, i de vegades detecta algunes direccions on es requereix millores de rendiment. Per un costat, ja s'ha començat a treballar amb els grups de l'OGC GMLJP2 i WCS per incorporar el JPEG2000 al WCS 2.0 d'una forma harmonitzada amb l'estàndard GMLJP2. D'altra banda, la recerca futura podria incloure tècniques per posar en pràctica algunes de les recomanacions a les IDE que s'indiquen en el capítol 2. A més, l'OGC ha detectat alguns problemes d'harmonització entre els diferents enfocaments de servidors de tessel·les que seran discutits als experiments d'interoperabilitat de l'OWS-9, on tenim la intenció de continuar amb la nostra recerca. Particularment important és la consolidació de conceptes com els identificadors de dades i metadades i les implementacions de RESTful per als sistemes ben organitzats com l'INSPIRE, així com en el context d'estructures menys organitzades, com ara el Global Earth Observation System of Systems (GEOSS), un tema que serà discutit en les activitats del proper AIP5. Hem passant d'una situació en què era difícil trobar dades, a una en que trobem una ingent quantitat d'informació geospacial a la xarxa, i per això, existeix la necessitat d'eines adients que desenvolupin nous conceptes que puguin ajudar als científics, i als usuaris de dades en general, a

explotar l'enorme potencial que el SIG a Internet ofereix. Per tal de filtrar millor les fonts d'informació, el poder ser informat de la qualitat de les dades i el poder comparar diferents fonts de dades serà cada vegada més important, un aspecte que considerem clau en la nostra recerca futura. Finalment, la computació en el núvol (*cloud computing*) podria complementar aquesta allau de dades, donant accés immediat a la capacitat computacional que permetria realitzar estudis sense precedents utilitzant la teledetecció i els sensors *in-situ* actualment disponibles, un tema on també tenim previst obrir una línia d'investigació.

8.B. Conclusions (English version)

This PhD studies generic aspects of implementing geospatial standards, and particularly in the map services area in the second part. It also presents some practical solutions for improving the standards and their implementations.

The SDI phenomenon has been growing continuously over the last two decades. A decentralized model based on adopting international standards and Internet technologies has been crucial for developing the second generation SDI. Nevertheless, there are several issues in SDI development that need to be solved. A relatively straightforward practical use case in which the Catalan SDI was used to study accessibility to healthcare centres showed how the SDI failed to provide information about the data available in the Catalan administration and also did not provide distributed analytical tools.

The performance of the SDI can be improved without reconceptualizing its principles, but rather by reinforcing or even refocusing some aspects of the current generation SDI. In chapter 2 there is a list of performance and usability aspects that can be improved, so that a balance can be achieved between the client portal gadgets and the server applications behind them. Collection of metadata and the search processes need to be improved. Service metadata have to be clarified so that data as well as tools that work in a distributed environment can be found easily. True data availability with default symbolization needs to be combined with standard Web services in a seamless environment that can be enriched with Volunteered Geographic Information (VGI). To demonstrate how easily VGI can help SDI development, we have provided an example of a project website that mixes classical standard protocols with modern web mash-up techniques, allowing volunteers to complete metadata catalogues with the user perspective, and therefore to fill in the provider information gaps.

To overcome their original limitations the SDI and distributed GIS can also be improved by applying the reformulated RESTful architecture, which reproduces the hypermap concept defined two decades ago. The resource-oriented architecture style hides the complexity of Web services and unifies data, metadata and data access services, reusing simple principles that have been used in the Web from the beginning. A network of dynamic hyperlinked geospatial resources is accessed using global URIs in a distributed environment that defines a single, global hypermap. The current SDI have implemented RPC-SOA style services based on OGC and ISO international standards for over 10 years, while mass market Web services and VGI communities mainly use RESTful services. The WWH concept reconnects these two worlds again and lowers the entry barrier, providing a harmonized environment, adopting the semantic Web. GEOSS and INSPIRE are only two examples of large geospatial data systems composed of a distributed collection of data servers and systems that use heterogeneous data access standards and protocols to disseminate scientific and commercial data. By adopting many different standards, both systems improve interoperability but still face some difficulties. INSPIRE attempts to overcome these difficulties by defining guidelines and implementation rules, narrowing the standard set to a common profile for discovery under in a common data access framework. Despite these efforts, none

of these systems have an integrated mechanism for upgrading and maintaining new data. Implementations of the WWH proposed in this paper provide a unified interface for discovering data, accessing data and services, and also interrelating knowledge by improving access to geospatial data and the results of scientific studies. Common functionalities, such as spatially or thematically filtering geospatial collections, generating maps, executing processes, etc., are also included in the RESTful approach, making data access services (such as WFS and WCS) unnecessary and integrating other Web service standards such as WMS, WMTS and WPS. A reference implementation is presented which demonstrates that only minor extensions and modifications to a state-of-the-art GIS software are necessary so that it can be included in the WWH. A corporate RESTful CGI application acts as a broker, connecting local applications and files to the WWH.

In particular, in the map services arena, the WMTS standard presented in this PhD thesis is a tiled map service designed with performance and scalability in mind. A WMTS server implementation must be able to return tiles quickly. To do so, the WMTS converts the continuous space into a discrete space defined by a limited number of predefined zoom levels (scales) and a regular tessellated space that assigns a triple index to each tile. A good way of achieving good performance is to use locally stored pre-rendered tiles that do not require any further image manipulation or geo-processing. Server developers decide whether pre-rendered tiles will be generated in a previous tile-preparation process or generated on the fly using a caching mechanism. The WMTS interface allows a client to receive three types of resources either in response to a resource request in the resource oriented architectural style or in response to an operation in the service oriented architectural style. These resources and operations are: a ServiceMetadata resource (GetCapabilities response) describing the abilities and information holdings of the specific server implementation; a tile resource (GetTile response) showing a fragment of a map representation of a layer; and a FeatureInfo resource (GetFeatureInfo response) providing information about the features located at a particular pixel of a tile map. WMTS have many similarities to other OGC Web Services (OWS), such as WMS, WFS, and WCS. The service is provided in three different protocols: KVP, SOAP and RESTful.

A possible implementation for WMTS is to use JPEG2000 images as an internal storage format. Even if JPEG2000 images have a codeblock structure for easily extracting a region of the image at any resolution, JPEG2000 needs to be formatted as a tiled JPEG2000 in order to have the fastest possible response. WMTS scale sets and tile matrices can be restricted using a small set of rules that allows the server to respond faster in common formats or in even in a small JPEG2000 file. Moreover, a corporation can use a single data storage with a collection of JPEG2000 files that can be used as an internal repository for WMTS, WCS, SOS and JPIP data services, and also avoid duplicity of data formats and provide a highly efficient compressed storage format.

This PhD thesis also analyzed the response time of several common map servers implementations. All the servers analyzed performed slower when the number of simultaneous clients is increased. A cluster of servers is a very good option for solving this situation: a group of computers is able to respond at the same time to different clients by assigning each client to a different computer in the group. The results show

that WMS servers that are not optimized for tile response perform the worst if clients request tiles from them. MapServer and GeoServer with minimum data configuration do not require a data preparation process but do not perform as well as the other services that require indexing methods, like MiraMon Map Server or Express Server. MapServer (based on C++ code) performs better than GeoServer (based on Java code) for single client requests, but GeoServer is surprisingly faster for concurrent simultaneous requests. The fastest WMS server tested was Express Server, which works with MsSID or JPEG2000 compressed images that are only 5% of the original uncompressed size. The fastest Web map tile server in the three cases assessed (concurrent, semi-concurrent and sequential requests) was GeoWebCache built over GeoServer.

Future work: The research in this PhD thesis develops certain theoretical and applied topics related to Internet GIS and standards for geospatial services, and highlights areas where performance needs to be improved. We have already started working with GMLJP2 and WCS OGC groups in order to incorporate JPEG2000 into WCS 2.0 in harmony with the GMLJP2 standard. In addition, future research could cover techniques for implementing some of the recommendations for improving SDI stated in chapter 2. Moreover, the OGC has detected some harmonization problems between different tile approaches that will be discussed in the OWS-9 interoperability experiment in which we plan to continue our research. It is particularly important to consolidate concepts like data and metadata identifiers and the RESTful implementations for well organized systems like INSPIRE as well as less organized structures such as the Global Earth Observation System of Systems (GEOSS). This subject will be discussed in the forthcoming AIP5. It is necessary to develop good tools that expand new concepts that can help scientific and general users to exploit the enormous potential that Internet GIS offers. We have moved from a situation in which it was difficult to find data to a situation of data overload. It is becoming more and more important to be able to report on data quality and compare different data sources. Finally, cloud computing could complement the data avalanche and provide easy access to computer power, thus allowing unprecedented studies to be performed using remote sensing and in situ sensors. We plan to research further into this area.

BIBLIOGRAFIA/REFERENCES

La bibliografia recollida aquí només és la bibliografia emprada en la redacció de la Introducció, el resum de resultats i les conclusions. Els capítols 2, 3, 4, 5, 6 i l'Annex I presenten la seva pròpia bibliografia. Això fa que d'aquesta bibliografia també es trobi als capítols indicats però no tota. Donat que la majoria de referències són en anglès, s'ha optat per redactar-les enterament en anglès.

Alameh N. (2003), Chaining Geographic Information Web Services, *IEEE Internet Computing*, 6 (18), 22–29.

Albrecht J. (1999), Geospatial information standards. A comparative study of approaches in the standardisation of geospatial information. *Computers and Geosciences* 25, 9–24

Antenucci J. C., Brown K., Croswell P. L. and Kevany M. J. (1991), Geographic Information Systems: A Guide to the Technology. *Springer*; 1 edition, 312 pp

Badard T. and Richard D. (2001), Using XML for the exchange of updating information between geographical information systems. *Computers, Environment and Urban Systems* 25, 17–31

Bai Y., Di L. and Wei Y. (2009), A taxonomy of geospatial services for global service discovery and interoperability. *Computers and Geosciences* 35, 783–790

Barclay T., Gray J., Ekblad S., Strand E.and Richter J. (2006), Designing and Building TerraService. *IEEE Internet Computing*. September 2006.

Baumann P. (2010), OGC Web Coverage Service (WCS) Standard – Core, Ver. 2.0, OGC 09-110r3. Available from: http://portal.opengeospatial.org/files/?artifact_id=41437 [Accessed 11 March 2012].

Bermúdez L. E., Masó J. and Capdevila J. (2012) Capitulo 20: Open Geospatial Consortium (OGC). Fundamentos de las IDE, *UPMPress*. Miguel A. Bernabé-Poveda y Carlos M. López-Vázquez (Eds.) Available in Annex I.

Bernard L., Craglia M., Gould M. and Kuhn W. (2005), Towards an SDI Research Agenda. 11 EC-GIS, June 28- July 1, 2005, Sardinia.

Bishr Y. (1998), Overcoming the semantic and other barriers to GIS interoperability. *International Journal of Geographical Information Science* 12 (4), 299–314

Boursier P. and Mainguenaud M. (1992), Spatial Query Languages; Extended SQL vs Visual Languages vs Hypermaps. *Proc 5th Int. Symposium on Spatial Data Handling*. Charlestopn VA, USA, August 1992, 242–295

Budhathoki N. R., Bruce B. C. and Nedovic-Budic Z. (2008), Reconceptualizing the role of the user of spatial data infrastructure. *GeoJournal* 72, 149–160.

Burt P.J. and Adelson E.H. (1983), The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, Com-31 (4).

Butler D. (2006), Virtual globes: the web-wide world. *Nature* 439, 776–778.

Butterfield M.L., Pearlman J.S. and Vickroy S.C. (2008), A System-of-Systems Engineering GEOSS; Architectural Approach. *IEEE Systems Journal* 2 (3), 321–332.

CEN 15449:2011: Standards, specifications, technical reports and guidelines required for SDI. *European Committee for Standardization*. 68 pp.

Chan T. O., Feeney M. E., Rajabifard A. and Williamson I. (2001), The dynamic nature of spatial data infrastructure: A method of descriptive classification, *Geomatica* 55 (1), 1–18.

Chow T.E. (2008), The Potential of Maps APIs for Internet GIS Applications. *Transactions in GIS* 12(2), 179–191.

Coleman D. and McLaulghlin J. D. (1997), Information access and network usage in the emerging spatial information marketplace. *Journal of Urban and Regional Information Systems Association* 9 (1), 8–19.

Craglia M., Goodchild M. F., Annoni A. Camara G., Gould M., Kuhn W., Mark D., Masser I., Maguire D., Liang S. and Parsons E. (2008), Next-Generation Digital Earth. A position paper from the Vespucci Initiative for the Advancement of Geographic Information Science. *International Journal of Spatial Data Infrastructures Research*, 3, 146–167.

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

de la Beaujardiere, J. (2004), OGC Web Map Service (WMS) Interface, Ver.1.3.0, OGC 03-109r1. Available from: http://portal.opengis.org/files/?artifact_id=5316 [Accessed 20 March 2010].

Díaz P., Masó J., Sevillano E., Ninyerola M., Zabala A., Serral I.and Pons X. (2012), Data Quality Analysis in the GEOSS Clearinghouse. *International Journal of Spatial Data Infrastructures Research*. Available from: http://ijsdir.jrc.ec.europa.eu/index.php/ijsdir/article/view/278/313 [Accessed 24 March 2012].

Doyle A. (1997), WWW Mapping Framework. *OpenGIS Project Document 97-007*. Wayland, Massachusetts: Open GIS Consortium.

Fielding R.T. (2000), Architectural styles and the design of network-based software architectures. *Thesis (PhD)*. University of California.

Fritz S., McCallum I., Schill C. , Perger C., Grillmayer R., Achard F. , Kraxner F. and Obersteiner M. (2009), Geo-Wiki.Org: The Use of Crowdsourcing to Improve Global Land Cover. *Remote Sensing*, 1 (3), 345-354.

Goodchild M.F. (2007), Citizens as Voluntary Sensors: Spatial Data Infrastructure in the World of Web 2.0. International Journal of Spatial Data Infrastructures Research, 2, 24–32.

Goodwin J., Hart G. and Dolbear C. (2008), Geographical Linked Data: The Administrative Geography of Great Britain on the Semantic Web. *Transactions in GIS*, 12(Suppl. 1), 19–30

Gregorio J., Fielding R., Hadley M., Nottingham M. and Orchard D. (2012), URI Template. Request for Comments: 6570. *Internet Engineering Task Force (IETF) Standards Track* ISSN: 2070-1721 Available from: http://www.rfc-editor.org/rfc/rfc6570.txt [Accessed 6 April 2012]

Gregorio J. (2008), URI Template. *Network Working Group Internet- Draft*. Available from: http://tools.ietf.org/html/draft-gregorio-uritemplate-03 [Accessed 14 September 2011][39].

Haklay M. and Weber P. (2008), OpenStreetMap: User-Generated Street Maps. *, IEEE Pervasive Computing*. 7 (4) 12–18.

Han H., Giles C. L., Manavoglu E., Zha H., Zhang Z. and Fox E. A. (2003), Automatic Document Metadata Extraction using Support Vector Machines. *3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, 37–48.

Heywood I., Cornelius S. and Carver S. (2006), An Introduction to Geographical Information Systems. Third Edition. *Pearson Prentice Hall*. 426 pp.

ISO/IEC, 2004, 15444-1:2004, Information technology – JPEG 2000 image coding system: Core coding system. Geneva, Switzerland, 194 pp.

ISO/IEC, 2004, 15444-9:2005, Information technology – JPEG 2000 image coding system: Interactivity tools, APIs and protocols. Geneva, Switzerland, 111 pp.

ISO/TC 211, 2003. ISO 19115:2003 Geographic Information – Metadata. Geneva, Switzerland, 146 pp.

Kalantari M., Olfat H. and Rajabifard A. (2010), Automatic Spatial Metadata Enrichment: Reducing Metadata Creation Burden through Spatial Folksonomies. *GSDI 12 World Conference*, Singapore. Available from: http://www.gsdi.org/gsdiconf/gsdi12/papers/23.pdf

Kim J.W., Park S.S., Kim C.S. and Lee Y. (2004), The Efficient Web-Based Mobile GIS Service System through Reduction of Digital Map. A. Laganà *et al.* (Eds.): ICCSA 2004, *Springer-Verlag LNCS* 3043, 410–417.

Kraak M.J. and Driel R.V. (1997), Principles of hypermaps. *Computers and Geosciences* 23 (4), 451–464.

Laurini R., and Milleret-Raffort F. (1990), Principles of geomatic hypermaps. *Proc. 4th Intern. Symposium on Spatial Data handling*, Zurich, 2, 642–655.

Manso M.A. and Wachowicz M. (2009), GIS Design; A Review of Current Issues in Interoperability. Geography Compass 3 (3), 1105–1124.

Manso-Callejo M., Wachowicz M. and Bernabé-Poveda M. (2009), Automatic Metadata Creation for Supporting Interoperability Levels of Spatial Data Infrastructures. *GSDI11 World Conference*. Rotterdam, The Netherlands, 15-19 June

Manso-Callejo M.A., Wachowicz M. and Bernabé-Poveda M. (2009b), Automatic Metadata Creation for Supporting Interoperability Levels of Spatial Data Infrastructures. *GSDI 11 World Conference*, Rotterdam, The Netherlands. Available from: http://www.gsdi.org/gsdiconf/gsdi11

[39] Aquesta referència ha quedat obsoleta donat que la molt recent (Geogorio et al., 2012) la reemplaça en el procés d'estandardització del IETF. S'ha conservat (Geogorio, 2008) perquè és la que en el seu dia va ser introduïda a la versió original del capítol 3 i del capítol 4.

Masó J., Pomakis K. and Julià N. (2010a), Web Map Tile Service Implementation Standard (WMTS). OGC 07-057r7. Open Geospatial Consortium. Available from: http://www.opengeospatial.org/standards/wmts and chapter 4

Masó J., Zabala A. and Pons X. (2010b), Combining JPEG2000 Compressed Formats and OGC Standards for Fast and Easy Dissemination of Large Satellite. Italian Journal of Remote Sensing (Rivista Italiana di Telerilevamento) 42 (3), 101–114. Available in chapter 5.

Masó J., Diaz P., Pons X., Monteagudo-Pereira J. L., Serra-Sagristà J. and Aulí-Llinàs F. (2011a), Impact of User Concurrency in Commonly Used Open Geospatial Consortium Map Server Implementations. *The International Conference on Advanced Communications and Computation (INFOCOMP 2011)*. Rückemann CP, Christmann W, Pankowska M. (Eds.), IARIA, Barcelona. 23 al 29 d'octubre. ISBN: 978-1-61208-161-8. Available in chapter 6

Masó J. , Pons X., Schäffer B., Foerster T. and Lucchi R. (2011b), Haiti Earthquake: Harmonizing post-event distributed data processing. *Earthzine* 3 (3). Available from: http://www.earthzine.org/2011/03/18/haiti-earthquake-harmonizing-post-event-distributed-data-processing. [Accessed March, 2012]

Masó J., Pons X. and Zabala A. (2012a), Tuning the second generation SDI: Theoretical aspects and real use cases. *International Journal of Geographical Information Science*. DOI:10.1080/13658816.2011.620570. Available in chapter 2.

Masó J., Pons X. and Zabala A. (2012b) Building the World Wide Hypermap (WWH) with a RESTful architecture. *International Journal of Digital Earth*. DOI:10.1080/17538947.2012.669414 Available in chapter3.

Mazzetti P., Nativi S. and Caron J. (2009), RESTful implementation of geospatial services for Earth and Space Science applications. *International Journal of Digital Earth* 2 (1), 40–61.

Meeksa W.L. and Dasguptab S. (2004), Geospatial information utility; an estimation of the relevance of geospatial information to users. *Decision Support Systems* 38, 47–63.

Mohammadi H., Rajabifard A. and Williamson I.P. (2010), Development of an interoperable tool to facilitate spatial data integration in the context of SDI. *International Journal of Geographical Information Science* 24 (4), 487–505.

Morris C. (2012), Exploring contemporary amateur meteorology through an historical lens. *Weather* 67 (1), 4–8.

Moshfeghi M. and Ta J. (2004), Efficient Image Browsing with JPEG2000 Internet Protocol. *Medical Imaging*: PACS and Imaging Informatics, Proceedings of SPIE, (5371), 31–42.

Muller M. (2006), Symbology Encoding (SE) Implementation Specification, Version 1.1.0, OGC 05-077r4. Available from: http://portal.opengeospatial.org/files/?artifact_id=16700 [Accessed 20 March 2010].

Muller M. and MacGill J. (2005), Styled Layer Descriptor Profile of the Web Map Service Implementation Specification, Version 1.1.0, OGC 05-078, Available from: http://portal.opengeospatial.org/files/?artifact_id=22364 [Accessed 20 March 2010].

Na A. and Priest M. (2007), OGC Sensor Observation Service SOS, Ver. 1.0.0 OGC 06-009r6, Available from: http://portal.opengeospatial.org/files/?artifact_id=26667 [Accessed 20 March 2010].

Nebert D., Whiteside A. and Vretanos P.P. (2007), OGC Catalogue Service Implementation Specification, Version 2.0.2, OGC 07-006r1, Available from: http://portal.opengeospatial.org/files/?artifact_id=20555 [Accessed 20 March 2010].

Nebert D.D. (2004), The SDI Cookbook. Available from: http://www.gsdi.org/docs2004/Cookbook/cookbookV2.0.pdf [Accessed 20 March 2010].

Nedovic-Budic Z. and Pinto J. K. (2000), Information sharing in an interorganizational GIS environment, *Environment and Planning B: Planning and Design* 27, 455–474.

Olfat H., Rajabifard A., and Kalantari M. (2010), Automatic Spatial Metadata Update: a New Approach. *XXIV FIG International Congress*, Sidney Australia. Available from: http://www.fig.net/pub/fig2010/papers/ts05b/ts05b_olfat_rajabifard_et_al_4079.pdf

Olivet M., Aloya J., Prat E. and Pons X. (2008), Health services provision and geographic accessibility. *Medicina Clinica*, 131 (4), 16–22.

Pascual V, Guimet J, Szczerban W. and Corcoll S. (2009), CatalogConnector: An OGC CSW client to connect metadata catalogues. *GSDI 11 World Conference*, Rotterdam, The Netherlands. Available from: http://www.gsdi.org/gsdiconf/gsdi11/

Patel S., Yuval S. and Lau D. (2007), Interactive Image Browsing with JPEG2000/JPIP. *Master project in Image Communication I*. Stanford University, Available from: http://www.stanford.edu/class/ee398a/projects/reports/LauPatelYuval.pdf [Accessed 26 November 2009].

Paudyal D.R., McDougall K. and Apan A. (2009), Building SDI Bridges for Catchment Management. In: Loenen B., Besemer J.W.J. and Zevenbergen J.A., *SDI Convergence*, 265–279.

Peng Z.R., and Tsou M.H. (2003), Internet GIS. Distributed Geographic Information Services for the Internet and Wireless Networkds. *John Wiley and Sons Inc*. Hoboken, New Jervey USA.

Percivall G. (2010), The application of open standards to enhance the interoperability of geoscience information, *International Journal of Digital Earth*, 3(S1), 14–30.

Pons X. (1996), Els Sistemes d'Informació Geogràfica: la nova carta. *Butlletí de l'Institut Català Història Natural* 64, 37–52.

Pons X. (2002), MiraMon. Geographic Information System and Remote Sensing software, *Centre de Recerca Ecològica i Aplicacions Forestals*, CREAF. Bellaterra. ISBN: 84-931323-5-7.

Pons X. and Masó J. (2000), MiraMon Map Reader, a new way for distributing and exploring geographic information. XIX Congress of the International Association for Photogrammetry and Remote Sensing (ISPRS). Amsterdam.

Putz S (1994), Interactive Information Services Usign World Wide Web Hypertext. *Computer Networks and ISDN Systems* 27 (2) November, 273–280.

Quinn S. and Gahegan M. (2010), A Predictive Model for Frequently Viewed Tiles in a Web Map. *Transactions in GIS* 14(2), 193–216.

Scholten M., Klamma R. and Kiehle C. (2006), Evaluating Performance in Spatial Data Infrastructures for Geoprocessing. *IEEE Internet Computing*, October, 34–41.

Schut P. (2007), OGC Web Processing Service (WPS), Version 1.0.0, OGC 05-007r7, Available from: http://portal.opengeospatial.org/files/?artifact_id=24151 [Accessed 20 March 2010].

Sheppard S.R.J.and Cizek P. (2009), The ethics of Google Earth; Crossing thresholds from spatial data to landscape visualisation. *Journal of Environmental Management* 90, 2102–2117.

Tu S. Flanagin M., Wu Y., Abdelguerfi M., Normand E. and Mahadevan V. (2004), Design Strategies to Improve Performance of GIS Web Services. *Proceedings of the International Conference on Information Technology: Coding and Computing* 444–450.

van Loenen B. (2009), Developing geographic information infrastructures; the role of access policies. *Journal of Geographical Information Science* 23(2),195–212.

Vandenbroucke D. (2008), Spatial Data Infrastructures in Europe: State of Play 2007. K.U.Leuven (SADL + ICRI).

Vanmeulebrouk B., Bulens J., Krause A. and Groot H. (2009), OGC standards in daily practice; gaps and difficulties found in their use. *GSDI11 World Conference*, Rotterdam, The Netherlands, 15-19 June.

Voisard, A. (1998), Geologic hypermaps are more than clickable maps. *Proceedings of the 6th ACM international symposium on Advances in GIS*. Washington, D.C., United States, 14–19.

Vretanos P.A. (2010), OGC Web Feature Service (WFS) Interface Standard, Ver. 2. 0, OGC 00-025r1, Available from: http://portal.opengeospatial.org/files/?artifact_id=39967 [Accessed 15 Abril 2012].

Wang S. and Liu Y. (2009), TeraGrid GIScience Gateway; Bridging cyberinfrastructure and GIScience. *International Journal of Geographical Information Science* 23 (2),169–193.

Wick M. and Becker T. (2007), Enhancing RSS Feeds with Extracted Geospatial Information for Further Processing and Visualization. In: Scharl A. and Tochtermann K. eds. *The Geospatial* Web*. How Geobrowsers, Social Software and the* Web *2.0 are Shaping the Network Society*. Berlin, Springer. 105–115. ISBN: 978-1-84628-827-2

Wright D. J., O'Dea E., Cushing J. B., Cuny J. E. and Toomey D. R., (2003), Why Web GIS May Not Be Enough; A Case Study with the Virtual Research Vessel. *Marine Geodesy* 26 (1), 73–86.

Yang C., Goodchild M., Huang Q., Nebert D., Raskin R., Xu Y., Bambacus M. and Fay D. (2011), Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing. *International Journal of Digital Earth* 4 (4), 305–329

Yang C., Wong D. W., Yang R., Kafatos M. and Li Q. (2005), Performance-improving techniques in Web-based GIS. *International Journal of Geographical Information Systems* 19 (3), 319–342.

Yuan X., Chen N., and Gong J. (2000), A distributed hypermap model for Internet GIS, *Geo-Spatial Information Science*. 3(4), 9–15.

Zabala A. (2010) Lossy Compression Effects on Remote Sensing Images and Resulting Cartography, Doctorat en Geografia. Universitat Autonòma de Barcelona. *PhD Tesis*, 168 pp.

Zabala A. and Masó J. (2005), Integrated hierarchical metadata proposal: series, layer, entities and attributes. *Proceedings of International Cartographic Conference*, 9-16 July, A Coruña.

zur Muehlen M., Nickerson J.V. and Swenson K.D. (2005), Developing web services choreography standards—the case of REST vs. SOAP. *Decision Support Systems* 40, 9–29.

ANNEX I: FUNDAMENTOS DE LAS IDE. OPEN GEOSPATIAL CONSORTIUM (OGC)

Aquest capítol és una reproducció del text enviat com a capítol pel llibre “Fundamentos de las IDE”, acceptat per publicar aquest any 2012: Bermúdez L. E., Masó J. , Capdevila J. (2012) Capitulo 20: Open Geospatial Consortium (OGC) al llibre "Fundamentos de las IDE", UPMPress Miguel A. Bernabé-Poveda y Carlos M. López-Vázquez (Eds.) (en aquest document es referencia com Bermúdez *et al.*, 2012)

Open Geospatial Consortium (OGC)

Luis E. Bermúdez[1], Joan Masó[2], Joan Capdevila[3]
[1]Open Geospatial Consortium, Herndon, VA, USA
[2]Centre de Recerca Ecològica i Aplicacions Forestals (CREAF),
Universidad Autónoma de Barcelona, España
[3]Instituto Geográfico Nacional, Servicio Regional en Cataluña, Barcelona, España
[1]lbermudez@opengeospatial.org, [2]joan.maso@uab.es, [3]joan.capdevila@mpt.es

Resumen. La organización internacional OGC está formada por agencias gubernamentales, universidades, compañías y centros de investigación, y tiene como misión promover el uso de estándares y tecnologías abiertas en el área de sistemas y tecnologías de la información geográfica y afines. El OGC mantiene el programa de especificación de estándares, el programa de experimentación en interoperabilidad y el programa de adopción. Dentro del programa de especificación, los grupos de trabajo de estándares elaboran, por consenso, documentos que estandarizan aspectos independientes, mientras que los grupos de trabajo temáticos debaten como mejorar la interoperabilidad de información geoespacial en sectores profesionales determinados o para temas concretos. A pesar del interés inicial por los estándares para API que fueron desarrollados por el OGC hace ya más de una década, la aparición de las IDE, y su necesidad de establecer plataformas interoperables y distribuidas en la web, ha potenciado el éxito de los estándares para servicios web que usen, en lo posible, estándares de codificación y de datos. Los servicios web tienen una estructura cliente–servidor común basada en el uso que protocolos de comunicación con peticiones codificadas sobre una URL (a partir de pares clave-valor KVP) o en documentos XML o SOAP. Todos los servicios comparten una petición común denominada *GetCapabilities* y un mecanismo de negociación de versiones. La respuesta a estas peticiones depende del tipo de servicio pero comúnmente se trata de un documento codificado en un dialecto específico de XML. Así, estos pueden clasificarse en servicios de visualización (como los servicios que generan mapas: WMS; o que responden mapas fragmentados en teselas: WMTS, completados por las extensiones de simbolización: SLD y SE; y el servicio de enlace de tablas: TJS), de accesos a datos (ya sean datos vectoriales, *features*: WFS; sobre datos ráster, *coverages*: WCS; o provenientes de observaciones de sensores, *sensor observations*: SOS), de codificación de datos (para visualización sobre globos virtuales: KML; para datos georeferenciados vectoriales GML; para observaciones y medidas de sensores: O&M; o para descripción de los propios sensores: SensorML), catálogo (CSW, en sus diversos perfiles entre los que destaca el ISO19115) y procesos (genéricos: WPS, o específicos para datos ráster WCPS), todo ello dentro del marco de referencia de OGC que determina la relación entre los diferentes estándares tanto los abstractos, los de implementación y los perfiles. En el desarrollo de IDE, los estándares y el proceso de OGC permiten establecer plataformas interoperables y distribuidas en la web. Esto posibilita que los países y comunidades del mundo puedan tomar mejores decisiones, por ejemplo para prevenir o paliar los efectos de catástrofes o para planificar mejor el desarrollo de comunidades.

Palabras Clave: *Software* Abierto, Ingeniería Web, OGC, Interoperabilidad

20.1 Introducción

El desarrollo de Internet ha permitido que organizaciones de todo el mundo puedan conectarse a una gran red de comunicación y publicar información a la que otros pueden acceder y utilizar. Se han desarrollado varias infraestructuras basadas en Internet, como es el caso de las IDE, nombre con el que se conocen a las plataformas interoperables y distribuidas en la web que permiten la integración de información geográfica. Las IDE, al igual que otras infraestructuras, están basadas en acuerdos entre sus miembros que, entre otras cosas, concretan el uso de tecnologías para publicar, acceder, visualizar y procesar los datos por Internet.

El OGC tiene como misión promover el uso de estándares y tecnologías abiertas en el área de sistemas y tecnologías de la información geográfica y áreas afines. OGC, a través de sus programas y tecnologías de colaboración, hace posible que organizaciones, incluyendo las agencias gubernamentales, universidades y el sector privado, lleguen a acuerdos para desarrollar sistemas y herramientas de información geográficas basados en tecnologías abiertas.

Este capítulo describe la visión y propuesta tecnológica de OGC, incluyendo los programas de especificación y de interoperabilidad, para el estímulo y desarrollo tanto de las IDE como de otras estrategias de intercambio de IG en sistemas distribuidos. Los estándares más importantes de visualización, de acceso de datos, de codificación de datos, de catálogos y registro, y de procesado de datos se explican brevemente dentro del marco de referencia de OGC.

20.2 El *Open Geospatial Consortium*

El OGC es una organización que tiene como misión promover el uso de estándares y tecnologías abiertas en el área de sistemas y tecnologías de la información geográfica y áreas afines.

En 2011, OGC agrupa ya más de 400 organizaciones incluyendo agencias gubernamentales, universidades, compañías y centros de investigación que pretenden colaborar con el desarrollo de especificaciones y estándares. Actualmente el Comité técnico del OGC promueve la creación de grupos de trabajo, constituidos en su gran mayoría por voluntarios de las organizaciones miembros, que se rigen mediante procesos de consenso. Los resultados de los grupos de trabajo se traducen en estándares abiertos y públicos que permiten soluciones interoperables que facilitan el acceso, la manipulación y el intercambio de información geoespacial en la web.

Las actividades del OGC se articulan alrededor de tres programas: a) Programa de especificación, b) Programa de interoperabilidad y c) Programa de alcance y adopción.

El consorcio cuenta con el personal y la tecnología necesaria para dinamizar la actividad de todos los involucrados, formando una infraestructura de colaboración que

facilita el trabajo en equipo, inclusive cuando sus miembros se hallen dispersos por todo el mundo. Entre las tecnologías se encuentra un sistema de gestión de contenidos en la web, foros, infraestructura para teleconferencia y listas de correo. El Comité técnico del OGC se reúne físicamente cuatro veces al año en diferentes partes del mundo. EL OGC se financia con las cuotas de sus miembros, licencias de certificación y con los ingresos generados por la gestión de proyectos, que forman parte del programa de interoperabilidad.

20.2.1 El programa de especificación

El programa de especificaciones (*Specifications Program*, SP) de OGC define el proceso mediante el cual se formaliza un estándar OGC. Se caracteriza por abordar el problema de la interoperabilidad desde un punto de vista teórico y deliberativo, donde los resultados deben aprobarse de forma consensuada por parte de los miembros de OGC. El eje coordinador de este programa es el Comité técnico (*Technical Committee*, TC), el cual decide qué trabajos se van a emprender y sobre la validez de los resultados.

El punto de partida para la elaboración de un estándar OGC puede ser una propuesta específica llevada a cabo por un miembro, basada en la detección de una carencia o necesidad a partir de los resultados del Programa de interoperabilidad. El TC forma un grupo de trabajo de estándares (SWG) *ad-hoc* para cada propuesta. El SWG es el encargado de concretar la propuesta, siguiendo las pautas establecidas por el conjunto de documentos de la Especificación Abstracta (*Abstract Specification*), un modelo para el desarrollo de especificaciones establecido por OGC. El resultado es una serie de documentos técnicos y propuestas de acciones. El TC es el marco donde los miembros de OGC consideran las propuestas de los SWG y donde, entre otras cosas, se recomienda la adopción de estándares OGC.

El TC también puede promover la creación de grupos de trabajo temáticos (DWG) donde se discuten las cuestiones relacionadas con la interoperabilidad de información geoespacial en sectores profesionales determinados o para temas concretos. Los resultados del trabajo de los DWG también deben formalizarse según la tipología OGC y las propuestas deben elevarse al TC para su consideración.

La última responsabilidad la ostenta el Comité de planificación, quien tiene capacidad de aprobar y adoptar las recomendaciones del TC y hacerlas públicas como estándar OGC.

20.2.2 El Programa de interoperabilidad

El Programa de interoperabilidad de OGC permite definir, documentar, mejorar y poner a prueba las especificaciones actuales de OGC. Uno de los propósitos principales de este programa, es conseguir de manera rápida y práctica (basados en esfuerzos de colaboración), el desarrollo de estudios y prototipos de infraestructuras interoperables basados en estándares OGC y especificaciones presentados por los participantes de los proyectos. El OGC también proporciona un programa de servicio

de pruebas de conformidad (CITE) que permite evaluar y certificar la compatibilidad de una aplicación con un estándar determinado (Percivall, 2010).

20.3 Introducción a los estándares de OGC

El principal producto o resultado de los programas OGC son los estándares materializados en especificaciones, es decir, en documentos que detallan interfaces informáticas o formas de codificación de datos. Cada estándar se piensa para solucionar aspectos específicos de interoperabilidad, de manera que su implementación en productos y servicios produzca resultados independientes del productor y de la implementación. En la aproximación cliente-servidor, los estándares deben permitir que los desarrollos informáticos a cada lado puedan intercambiar datos sin necesidad de adecuar los correspondientes códigos.

OGC trabaja también en términos de estándar abierto, lo que significa que deben estar disponibles de forma pública y libre, sin discriminación alguna, sin costes, independiente de quien lo proporciona o de los datos que maneja y aprobado formalmente mediante consenso.

La elaboración de estándares debe llevarse a cabo de forma coordinada y coherente con el estado de desarrollo ya conseguido dentro de OGC. El **modelo de referencia OGC** (Percivall *et al.*, 2008) es un documento que describe las relaciones entre los documentos considerados básicos, es decir, los documentos de estándares abstractos y de implementación (interfaz, codificación, perfil o esquema de aplicación), y las buenas prácticas. El modelo de referencia se actualiza periódicamente y proporciona una visión general de los resultados del trabajo de los miembros de OGC que han contribuido a esa documentación básica.

20.3.1 Clasificación de los estándares OGC

El OGC produce y revisa estándares desde hace más de dos décadas. De manera muy general, los estándares OGC pueden agruparse en cuatro grandes categorías: estándares de codificación y datos, estándares de servicios web, estándares para API y estándares para clientes web (fig. 1).

20.3.2 Arquitectura de los Servicios OGC

Los servicios web de OGC comparten un conjunto de características comunes que se han recogido en el estándar *OWS Common* (Whiteside y Greenwood, 2010).. La mayoría de estándares de servicio han seguido o están siguiendo un proceso de armonización con *OWS Common*. Las principales características comunes de los entandares de servicios OGC son:

- Arquitectura cliente-servidor. Los estándares de servicio separan las aplicaciones en dos partes: el cliente y el servidor. Se establece un protocolo de comunicaciones entre ambos basado en operaciones solicitadas por el cliente, y respondidas por el servidor (llamadas a procedimiento remoto: RPC). Así, cada

estándar de servicio define un conjunto limitado de operaciones para las que se define la petición y la respuesta.

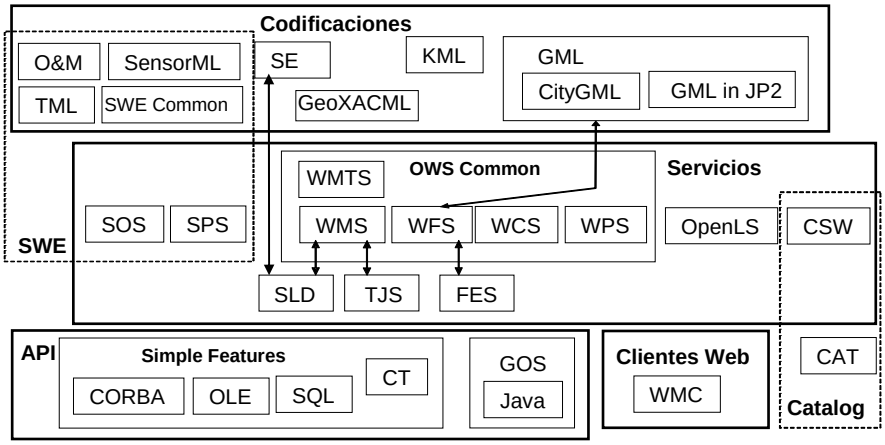
- 

Figura 1. Clasificación de los estándares OGC (Fuente: creación propia de Joan Masó).

- Protocolos web. En general, los servicios adoptan el protocolo de comunicaciones web, en el que las peticiones se efectúan a partir de una URL y la respuesta a tales peticiones es un documento (generalmente en formato XML).
- Los protocolos web utilizados como base para las operaciones son los pares clave valor (KVP) en la URL (petición HTTP GET) y las peticiones en codificación XML (petición HTTP POST). Recientemente se ha adoptado el protocolo SOAP y algunos servicios introducen el estilo de arquitectura REST.
- Formato de documento. Las peticiones envían documentos y los solicitan en un formato de documento de respuesta, lo que se conoce como tipos MIME (MIME *types*), como por ejemplo “text/xml”.
- Mecanismo de control de errores (excepciones). Cuando se produce un error, en general, la respuesta no es el documento esperado sino un documento excepcional que describe la característica del error.
- Existe una operación común, y obligatoria, para todos los servicios OGC denominada *GetCapabilities*, que permite conocer a los clientes las características generales del servicio, a partir de un conjunto de metadatos de servicio (*service metadata*) y de las operaciones admitidas (*capabilities*). La respuesta tiene una parte común (metadatos de identificación de servicio, de proveedor de servicio, de descripción de operaciones, y listado de idiomas disponibles), y una parte que depende de cada tipo de servicio y que describe los recursos que el servicio pone a disposición. [1]

[1] La palabra "recurso" tiene aquí un significado deliberadamente ambiguo. En los estándares de visualización y acceso a datos se la puede asociar con "conjunto de datos" (*dataset*). Sin embargo, el "conjunto de datos" recibirá un nombre distinto en cada uno de los servicios para

- Negociación de versiones. Todos los servicios comparten un sistema de negociación de versiones.

En un entorno web es posible descubrir información, valorar su utilidad, acceder a estos datos y procesarlos. En los siguientes apartados, se presenta una clasificación de estándares para servicios espaciales del OGC en cuatro tipos según su propósito: visualización, acceso a datos, catálogo y procesado. Todos los servicios descritos a continuación incorporan y describen la petición *GetCapabilities* que no se describe individualmente en favor de la claridad y brevedad del texto.

20.4 Estándares OGC para visualizar datos

Estos estándares se aplicaron originalmente a la creación de portales que visualizaban mapas (representación en forma de imagen cartográfica con la resolución necesaria para mostrarse en el dispositivo de salida) incrustados en páginas web. Sin embargo, sus aplicaciones se han extendido a todo tipo de productos, desde SIG de escritorio a aplicaciones para dispositivos móviles.

El estándar WMS (de la Beaujardiere, 2004) define una operación obligatoria (*GetMap*) para obtener un mapa de una zona definida por su ámbito y dimensiones en píxeles a partir de los datos de una o varias capas (*layers*). Generalmente, esta representación se codifica en un formato común en los navegadores de mapas (image/jpeg o image/png). Los estilos de simbolización están predefinidos en el servidor.

Adicionalmente proporciona una operación opcional (*GetFeatureInfo*) que permite obtener más información sobre un punto del mapa, generalmente en un formato de texto. Se usa normalmente para implementar la consulta por localización.

Los estándares SLD (Lupp, 2007) y SE (Müller, 2006) permiten al usuario definir nuevos estilos de simbolización. SLD es una extensión del WMS que describe ampliaciones de la petición *GetMap* para solicitar estilos definidos por el usuario a un servidor WMS. SE es un lenguaje transversal de codificación de estilos, codificado en XML y aplicable tanto al modelo raster como al vectorial.

El estándar TJS (Schut, 2010) permite al usuario enriquecer los mapas disponibles en un servicio WMS a partir de tablas de información que contienen nuevos atributos. Define un formato XML para tablas de datos denominado GDAS, un conjunto de operaciones que permiten describir y obtener las tablas de datos disponibles por el servicio (*DescribeFrameworks*, *DescribeDatasets*, *DescribeData* y *GetData*), y un conjunto de operaciones que permiten enlazar una tabla GDAS con un conjunto de entidades espaciales y crear una nueva capa un servidor WMS asociado (*DescribeJoinAbilities*, *DescribeKey* y *JoinData*).

El estándar WMTS (Masó *et al.*, 2005) es muy similar al WMS, excepto que discretiza el espacio en un conjunto de niveles de zoom predefinidos y para cada uno de ellos define una matriz regular de teselas. Las teselas son indivisibles y sólo se pueden obtener una por una a partir de la petición *GetTile*. Ello permite sacar

evidenciar su distinta naturaleza (*layer* en WMS, *coverage* en WCS, *feature type* en WFS, *process* en WPS, etc).

provecho de los mecanismos de caché de servidor que existen actualmente en la web, por lo que las interacciones con servidores utilizadas muy frecuentemente deben resultar más ágiles que los servidores WMS en igualdad de condiciones. Sin embargo, los clientes WMTS resultan un poco más complejos dado que, generalmente, deben realizar diversas peticiones de teselas adyacentes hasta llenar el área de navegación con cada acción del usuario. Este estándar incorpora una codificación REST basada en plantillas URL así como peticiones *GetFeatureInfo*.

20.5 Estándares OGC para el acceso a datos

En este grupo es posible distinguir los estándares que trabajan con datos en el modelo de datos vectorial y los que trabajan con datos en el modelo de datos raster. La mayoría de ellos sólo permiten la descarga de datos, mientras que algunos estándares (denominados transaccionales) permiten la actualización de datos.

El estándar WFS (Vretanos, 2005) permite el acceso a datos vectoriales en formato GML. Ofrece una operación para obtener el esquema de aplicación de una o varias *FeatureTypes* (*DescribeFeature*) y operaciones para descargar un documento GML (o una parte de él), con los datos vectoriales (*GetFeature*, *GetFeatureWithLock* y *GetGMLObject*). Adicionalmente, el estándar ofrece la capacidad transaccional de crear, editar y borrar entidades vectoriales (*Transaction*) del servidor, que pasan a estar disponibles inmediatamente para el resto de los usuarios.

El estándar WFS sólo permite filtrar las entidades vectoriales por identificador, por versión y por envolvente espacial. Si se desea mayor capacidad de filtrado, FE describe una codificación XML para limitar los valores de determinadas propiedades de las entidades (p. ej., número de habitantes superior a 1000). Se permite así, implementar una consulta por atributos. Más aún, permite también realizar filtros con criterios espaciales y topológicos como: la entidad contiene, cruza o está a una distancia inferior al rectángulo de la envolvente espacial o de un objeto.

El estándar **WCS** (Whiteside y Evans, 2008) permite acceder a datos raster. La principal característica del WCS es que considera los raster multidimensionales (p. ej. resultados de modelos climatológicos). WCS define 2 operaciones: una permite recuperar una descripción de cada una de las capas y todas sus dimensiones (*DescribeCoverage*), y otra permite obtener un subconjunto de los datos de las capas escogiendo una envolvente espacial y un rango dimensional (*GetCoverage*). A diferencia del WFS, no establece un formato de datos concreto y se prevé la futura aprobación de extensiones de servicio que estandaricen formatos de respuesta concretos.

El estándar **SOS** (Na y Priest, 2007) define la interfase para consultar y pedir información acerca de sensores y sus observaciones. Ayuda a sistemas de observación, como observatorios marinos, a establecer un servicio web con toda la información acerca de las plataformas y sus sensores, las observaciones en tiempo real, así como histórica, y datos brutos o procesados. Las dos operaciones más importantes de SOS son *DescribeSensor* y *GetObservation.*

20.6 Estándares OGC para codificación de datos

La codificación estandarizada de datos se ha controlado tradicionalmente por los propios fabricantes de *software* que imponían sus estándares *de facto*. La mayoría de estándares OGC utilizan como base de codificación el estándar de W3C XML que define un formato de texto controlado por un documento de esquema XSD.

KML (Wilson, 2008) es un ejemplo de estándar desarrollado fuera de OGC inicialmente por la empresa Keyhole y después por Google. KML es la codificación XML nativa de Google Earth, inherentemente 3D y pensado para la presentación final de datos (incluyendo incluso la posición del observador) y no para su almacenado y análisis. Un esquema XSD prefijado permite la definición de entidades vectoriales 3D con estilos de simbolización y un atributo HTML. Tiene limitaciones importantes como: la ausencia de definición de tipos de entidad, el soporte sólo a la proyección latitud-longitud en WGS84 y la inclusión de un único atributo para cada entidad.

El GML (Portele, 2007) es una codificación XML inicialmente concebida para el intercambio de datos vectoriales. El GML proporciona herramientas para la definición de entidades vectoriales 2D y 3D, simples o complejas. En la versión 3 incorpora también definiciones para el modelo raster pero su uso ha tenido poco éxito. Su gran flexibilidad es su punto fuerte y su debilidad. Un conjunto de datos vectoriales codificado en GML requiere la definición previa de los tipos de entidades a partir de propiedades espaciales definidas por el GML y de atributos temáticos definidos con tipos básicos XML o por tipos complejos personalizables. Los datos GML deben cumplir y validarse con el esquema de aplicación definido. En la práctica, esto significa que la estructura de un documento GML genérico resulta impredecible por las aplicaciones que deben leerlo, lo que se soluciona a partir de perfiles de aplicación GML.

El **perfil GML para entidades simples** (GML-SF) (Brink *et al.*, 2010) proporciona un conjunto de restricciones a la flexibilidad del GML y facilitan su uso práctico. Establece tres niveles de restricciones (SF-0, SF-1 y SF-2). Por ejemplo, el nivel más restrictivo (SF-0) limita las entidades gráficas a una lista de primitivas que permiten definir puntos, líneas o polígonos y los tipos de atributos a tipos XML básicos y la cardinalidad a 1. Una alternativa al uso del perfil genérico para entidades simples es el uso de **perfiles de aplicación** específicos para aplicaciones concretas; éste es el enfoque elegido por los perfiles de datos de directivas como INSPIRE.

Los estándares para sensores definen tres codificaciones de datos: SWE *Common*, O&M y *SensorML*. El estándar SWE *Common* (Robin, 2011) define el modelo conceptual y la codificación XML para la descripción de los datos relacionados con sensores. SWE *Common* logra la interoperabilidad sintáctica y semántica, para que los datos de sensores puedan ser entendidos por máquinas, procesados en flujos de trabajo y compartidos en nodos de sensores web.

El estándar O&M (Cox, 2011) define un modelo conceptual que determina que una medición es la acción de estimar el resultado de una propiedad de una entidad de interés (p. ej. temperatura del aire del centro de Barcelona) de acuerdo a un proceso (p. ej. lecturas por medio de termómetros). O&M también define una codificación XML para resultados de observaciones y está concebido para que se puedan desarrollar extensiones. Por ejemplo, SWE *Common* permite representar el resultado de la medición y *SensorML* (Botts y Robin, 2007) permite representar la descripción

de procesos (sensores, plataformas u otros procesos). Algunos de los aspectos que se pueden describir con *SensorML* son: posición geográfica, contacto, enlace con manuales, tipo de datos que miden, unidades de medida, control de calidad, procedencia, descripción sistemática y otros parámetros asociados a un procedimiento de medida.

20.7 Estándares OGC para catálogos y registros

Los servicios de catálogo ofrecen la capacidad de publicar y localizar colecciones de metadatos, tanto sobre datos como sobre servicios, así como sobre otras fuentes de información. Los metadatos recogidos y ofrecidos por estos catálogos contienen información que describen los recursos disponibles (fig. 2).

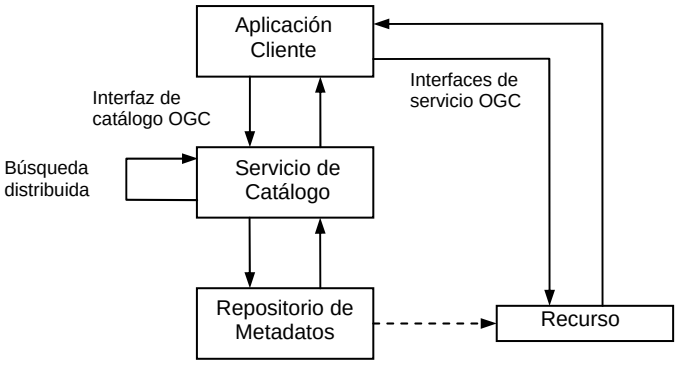


Figura 2. Arquitectura de referencia de OGC para servicios de catálogo.
(Fuente: Nebert *et al.*, 2007)

Esta información debe ser suficiente como para que se pueda interrogar a los recursos y ponerlos a disposición para su evaluación y proceso, tanto por humanos como por máquinas. Los servicios de catálogo son, por lo tanto, necesarios para la localización de recursos geoespaciales registrados dentro de una comunidad. Su papel en las IDE es esencial, dado que requiere de una capacidad de búsqueda de la información con cierto nivel de inteligencia, lo que a su vez exige que los metadatos estén basados en modelos bien conocidos que incluyan referencias espaciales. Sólo de esta manera se consigue automatizar y mejorar la eficiencia de la búsqueda de recursos geográficos (p. ej. otros servicios de OGC) (Sánchez Maganto, 2009).

La especificación del servicio de catálogo esta diseñada para dar soporte a varios niveles de detalle (Nogueras-Iso *et al.*, 2005). Por ejemplo, el perfil CSW 2.0.2 / ISO 19115, es un modelo que extiende el modelo general pero utiliza los modelos de metadatos ISO 19115/19139 sobre http (Nebert *et al.*, 2007). Este perfil es el más usado para la construcción europea de IDE.

20.8 Estándares OGC para el procesado de datos

Los estándares OGC hacen posible que la información espacial pueda distribuirse en Internet. Bajo tales circunstancias, resulta de gran utilidad la capacidad de ejecutar procesos analíticos remotos que extiendan las capacidades de los servicios web de visualización y consulta.

El estándar WPS (Schut, 2007) permite publicar, localizar y usar procesos geoespaciales. A través de la interfaz del WPS es posible describir los datos de entrada y de salida, así como los modos de ejecución y de acceso a la información producida. WPS permite que procesos específicos sobre datos geoespaciales (p. ej. interpolación sobre una rejilla determinada) se puedan publicar como servicios web para ser usados por clientes distribuidos en la web. Basado en WPS, el estándar WCPS se diseñó inicialmente como un estándar independiente. Este documento formaliza las operaciones de procesado raster y se está reformulando como un perfil WPS.

20.9 Conclusiones

El programa de interoperabilidad de OGC cuenta con iniciativas que ayudan a definir, documentar, mejorar y poner a prueba las especificaciones y estándares internacionales. El Programa de especificación define el proceso mediante el cual se formaliza un estándar OGC. Los estándares OGC son estándares públicos y abiertos, disponibles para todos sin coste alguno, independiente de quien lo proporcione o de los datos que maneje, y se aprueban formalmente mediante consenso.

Los estándares OGC se pueden categorizar en estándares de codificación y datos, de servicios web, de programación de aplicaciones (API) y para clientes web. En este capítulo, se han destacado los servicios web. Estos se basan en la arquitectura cliente-servidor, utilizan protocolos de la web, cuentan con mecanismos de control de errores y una operación común (*GetCapabilities*) que describe de manera consistente las características de cada servicio.

Al presente (fines de 2011) OGC tiene disponible una amplia lista de estándares, incluyendo WMS, SLD/SE, TJS y WMTS; WFS, FE, WCS, y SOS para acceso a datos; KML, GML, SWE Common, O&M y SensorML para codificación de datos; CSW con sus diferentes perfiles para catálogos y registros y finalmente WPS y WCPS para el procesado de datos.

20.10 Referencias

Botts M y Robin A (2007). *OpenGIS® Sensor Model Language (SensorML) Implementation Specification*, versión 1.0.0, OGC 07-000. (180 pp.). Recuperado el 8 de julio de 2011, de http://portal.opengeospatial.org/files/?artifact_id=21273

Brink L, Portele C y Vretanos, P. A. (2010). Geography Markup Language (GML) simple features profile (with technical note), versión 2.0.0, OGC 10-100r3. (87 pp.). Recuperado el 8 de julio de 2011, de http://portal.opengeospatial.org/files/?artifact_id=42729

Cox S. (2011). *Observations and Measurements - XML Implementation*, versión 2.0.0, OGC 10-025r1. (76 pp.). Recuperado el 8 de julio de 2011, de http://portal.opengeospatial.org/files/?artifact_id=41510

de La Beaujardiere, J. (2004). *OGC Web Map Service Interface*, version.1.3.0, OGC 03-109r1. (84 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengis.org/files/?artifact_id=4756

Lupp, M. (2007). *Styled Layer Descriptor Profile of the Web Map Service Implementation Specification*, versión 1.1.0, OGC 05-078r4. (53 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengeospatial.org/files/?artifact_id=22364

Masó, J., Pomakis, K. y Julià, N. (2005). *OpenGIS® Web Map Tile Service Implementation Standard*, versión 1.0.0, OGC 07-057r7. (128 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengeospatial.org/files/?artifact_id=35326

Müller, M. (2006). *Symbology Encoding Implementation Specification*, versión 1.1.0, OGC 05-077r4. (63 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengeospatial.org/files/?artifact_id=16700

Na, A. y Priest, M. (2007). *Sensor Observation Service*, versión 1.0.0, OGC 06-009r6. (104 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengeospatial.org/files/?artifact_id=26667

Nebert, D. Whiteside, A. y Vretanos, P. P. (2007). *OpenGIS® Catalogue Service Specification*, versión 2.0.2. OGC 07-006r1. (218 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengeospatial.org/files/?artifact_id=20555

Nogueras-Iso, J. Zarazaga-Soria, F.J., Béjar, R., Álvarez P.J. y Muro-Medrano, P.R. (2005). OGC Catalog Services: a key element for the development of Spatial Data Infrastructures. *Computers & Geosciences*, 31, 2, 199-209

Percivall G., Reed C., Leinenweber L., Tucker C. y Cary T (2008). *OGC Reference Model*, versión 2.0.0. OGC 08-062r4. (35 pp.). Recuperado el 15 de Junio de 2011, de http://portal.opengeospatial.org/files/?artifact_id=31112

Percivall, G. (2010). *OGC Standards - an overview tutorial*. CEODE. Beijing, China. (81 pp.). Recuperado el 15 de junio de 2011, de http://portal.opengeospatial.org/files/?artifact_id=41565

Portele C. (2007). *OpenGIS® Geography Markup Language (GML) Encoding Standard*, versión 3.2.1. OGC 07-036. (437 pp.). Recuperado el 9 de julio de 2011, de http://portal.opengeospatial.org/files/?artifact_id=20509

Robin A (2011) *OGC® SWE Common Data Model Encoding Standard*, versión 2.0.0, 08-094r1. (207 pp.). Recuperado el 8 de julio de 2011, de http://portal.opengeospatial.org/files/?artifact_id=41157

Sánchez Maganto, A. (2009). *Teoría CSW (Catalogue Service Web)*. (30 pp.). Recuperado el 9 de Junio de 2011, de http://www.slidefinder.net/c/csw/7818325

Schut, P. (2007). *OpenGIS® Web Processing Service*, versión 1.0.0, OGC 05-007r7. (87 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengeospatial.org/files/?artifact_id=24151

Schut, P. (2010). *OpenGIS® Georeferenced Table Joining Service (TJS) Implementation Standard*, versión 1.0.0, 10-070r2. (91 pp.). Recuperado el 8 de julio de 2011, de http://portal.opengeospatial.org/files/?artifact_id=40095

Vretanos, P. A. (2005). *Web Feature Service Implementation Specification*, versión 1.1.0, OGC 04-094. (121 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengeospatial.org/files/?artifact_id=8339

Whiteside, A. y Evans, J.D. (2008). *Web Coverage Service (WCS) Implementation Standard*, versión 1.1.2, OGC 07-067r5. (133 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengeospatial.org/files/?artifact_id=27297

Whiteside, A. y Greenwood, J. (2010). *OGC Web Services Common Standard*, versión 2.0.0, OGC 06-121r9. (207 pp.). Recuperado el 20 de marzo de 2011, de http://portal.opengeospatial.org/files/?artifact_id=38867

Wilson T. (2008). *OGC® KML*, versión 2.2.0, OGC 07-147r2. (251 pp.). Recuperado el 9 de julio 2011, de http://portal.opengeospatial.org/files/?artifact_id=27810

ANNEX II LIST OF RESOURCES AND OPERATIONS IN THE WWH

This Annex provides a comprehensive list of resource URIs and the meaning of the operations associated with each one in the WWH design. It was part of the materials for the chapter 3 but it was removed in the pair review phase due to page limitations imposed by editorial rules.

Resource	**URI template**
Supported methods and their meanings. ***Example***	
Organization	
Organization	http://{OrganizationServer}/WWH
GET: lists the software instances. POST: registers a new software instance. *HydroGIS, GeolGIS…*	
Organization metadata	http://{OrganizationServer}/WWH/_metadata
GET: retrieves metadata about the organization. PUT: updates metadata about the organization. *ACME corp…*	
Organization links	http://{OrganizationServer}/WWH/_link
GET: retrieves links to other related organizations. POST: adds a new link. *www.acme.ccc…*	
Organization link	http://{OrganizationServer}/WWH/_link/{link}
PUT: updates a link. DELETE: deletes a link.	
Software instance	
Software instance	http://{OrganizationServer}/WWH/{SoftwareInstance}
GET: lists the geospatial collections in this software instance. POST: adds a new geospatial collection. DELETE: removes the software instance. *Rivers, lakes…*	
Software instance metadata	http://{OrganizationServer}/WWH/{SoftwareInstance}/_metadata
GET: retrieves metadata about the software instance. PUT: updates metadata about the software instance. *The HydroGIS is responsible…*	
Geospatial collection	
Geospatial collection	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}
GET: lists the geospatial entities (in vector features), coverage collections (in raster coverages) or record sets (in databases). DELETE: removes the geospatial collection. *Amazon River, Mississippi River, Victoria Lake, Landsat multiband image…*	
Geospatial collection Metadata	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_metadata
GET: retrieves metadata common to the whole geospatial collection (including data type names such as: raster coverages, vector features, data tables, etc). PUT: updates metadata about the geospatial collection. *Rivers and lakes cover an extent of…*	
Geospatial collection links	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_link
GET: retrieves links to other related geospatial collections. POST: adds a new link. *http://www.fao.org/landandwater/aglw/sediment/sediment.xls…*	

Resource	**URI template**
Supported methods and their meanings. ***Example***	
Geospatial collection link	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_link/{link}
PUT: modifies a link. DELETE: deletes a link.	
Features	
Feature instance	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}
POST: creates a new feature instance	
Geospatial collection type	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_type
GET: retrieves a list of the data models in this geospatial collection. POST: adds a new _type for this data collection. *RiverType, LakeType…*	
Feature type	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_type/{type}
GET: retrieves the feature type definition. PUT: updates a feature type. DELETE: removes a feature type. *2d_line, name, length, sediment yield*	
Geospatial collection metadata	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_type/{type}/_metadata
GET: retrieves metadata of a particular feature type. PUT updates metadata. *All rivers cover an extent of…*	
Feature instance	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}
GET: retrieves a feature instance representation. PUT: updates a feature instance. DELETE: removes a feature instance. *x.y sequence, name, length and sediment yield of the Amazon River*	
Feature type	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}/_type
GET: retrieves the type of this feature. *RiverType*	
Feature collection filtered by type	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}?type={type}
GET: retrieves the feature collection (feature instances list) of a particular type. *Amazon River, Mississippi River …*	
Feature metadata	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}/_metadata
GET: retrieves metadata specific to this feature instance. PUT updates metadata specific to this feature instance. *Amazon River documented by…*	
Feature links	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}/_link
GET: retrieves a list of links to other features or feature collections related to this feature. POST adds a new link. *http://en.wikipedia.org/wiki/Amazon_River…*	
Feature link	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}/_link/{link}

Resource	URI template
Supported methods and their meanings. ***Example***	
PUT: modifies a link. DELETE: deletes a link.	
Feature attribute	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}/{attribute}
GET: retrieves the value of a particular attribute. PUT: updates the attribute value. DELETE: blanks the attribute value. *Amazon River sediment yield is…*	
Feature attribute links	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}/{attribute}/_link
GET: retrieves a list of links to attribute values. POST: adds a new link. Note that this acts as an external data value that could be in the linked data semantic web. Also note that geometries are also attributes and can be external. *http://www.fao.org/landandwater/aglw/sediment/default.asp?river=Amazon&search=Search*	
Feature attribute link	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}/{attribute}/_link/{link}
PUT: modifies a link. DELETE: deletes a link.	
Feature attribute type	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_type/{type}/{attribute}/_type
GET: retrieves the data model of this attribute. PUT: modifies the data model of this attribute. *Amazon River sediment yield is a floating point number*	
Feature attribute type metadata	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_type/{type}/{attribute}/_metadata
GET: retrieves metadata that is specific to the attribute type (and common to all attribute values). PUT updates metadata. *Amazon River sediment yield units: Mg/km²/yr*	
Feature attribute value metadata	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{GeospatialEntity}/{attribute}/_metadata
GET: retrieves metadata that is specific to the attribute in this particular feature. PUT: updates metadata. *Amazon River sediment yield measured by …*	
Feature attribute value meaning	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_all/{attribute}/{value}?type={type}
GET: retrieves additional information on the meaning of a value category of an attribute of a feature type. PUT: sets additional information. DELETE: deletes the additional information. *In a land cover attribute, a description of the " wetlands" category*	
Feature attribute value meaning link	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_all/{attribute}/{value}/_link?type={type}
GET: retrieves a list of links to the meaning of a value category. POST: adds a new link. Note that this acts as an external data value that could be in the linked data semantic web. *http://water.epa.gov/type/wetlands/types_index.cfm*	
Coverages	
Coverages	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}
POST: adds a new coverage to a coverage collection	
Coverage types	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_type
GET: retrieves the list of data models of the coverage collection.	

Resource	URI template
Supported methods and their meanings. ***Example***	
First band is a sequence of byte numbers representing…, second band is…	
Coverage	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_all
GET: retrieves the coverage collection. PUT: updates a coverage collection. DELETE: removes a coverage collection *Landsat multiband image*	
Coverage metadata	http://{OganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_all/_metadata
GET: retrieves metadata common to the coverage collection. PUT updates metadata. *Landsat multiband image owner: USGS*	
Coverage links	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_all/_link
GET: retrieves a list of links to other coverages or coverage collections. POST: adds a new link. *http://landsat.gsfc.nasa.gov/…*	
Coverage element	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}
GET: retrieves a coverage element. PUT: updates a coverage element. DELETE: removes a coverage *A band of a Landsat multiband image.*	
Coverage type	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}/_type
GET: retrieves the data model of this coverage element. PUT: updates the data model of this coverage element. *This band is a sequence of byte numbers representing…*	
Coverage metadata	http://{OganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}/_metadata
GET: retrieves metadata specific to a particular coverage element. PUT: updates metadata. *This band is sensing the spectral bandwidth: 0.63 - 0.69μm (red)*	
Coverage links	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}/_link
GET: retrieves a list of links to other coverages or coverage collections. POST: adds a new link.	
Coverage link	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}/_link/{link}
PUT: modifies a link. DELETE: deletes a link.	
Row values	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}/{row}
GET: retrieves a list of row values from a coverage element. PUT: sets a list of row values. DELETE: fills a row with NODATA values *The first row values of the red band of a Landsat multiband image are…*	
MultiRow values	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_all/{row}
GET: retrieves a bidimensional array of row values from a coverage collection. PUT: sets a row value array for a coverage collection. DELETE: sets a coverage collection row to NODATA *The first row values of all bands of a Landsat multiband image are…*	
Cell value	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollecti

Resource	URI template
Supported methods and their meanings. ***Example***	
	on}/{attribute}/{row}/{col}
GET: retrieves a coverage cell value. PUT: sets a coverage cell value. DELETE: sets coverage cell to NODATA *The first cell value of a band of a Landsat multiband image is…*	
Cell values	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/_all/{row}/{col}
GET: retrieves a coverage collection cell value array. PUT: sets a coverage collection cell value array. DELETE: sets a coverage collection cell to NODATA *The first cell values of all bands of a Landsat multiband image are …*	
Coverage value	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}/_all/_all/{value}
GET: retrieves additional information on the meaning of a value category. PUT: sets additional information. DELETE: deletes the additional information. Note: this is particularly useful when coverage values represent discrete categories. *In a land cover coverage, a string defining that category 5 means " wetlands"*	
Coverage value links	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}/_all/_all/{value}/_link
GET: retrieves a list of links to the meaning of a value category. POST: adds a new link. Note that this acts as an external data value that could be in the linked data semantic web. *http://water.epa.gov/type/wetlands/types_index.cfm*	
Coverage value link	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}/{attribute}/_all/_all/{value}/_link/{link}
PUT: modifies a link. DELETE: deletes a link.	
Cell values	http://{OrganizationServer}/WWH/{SoftwareInstance}/{GeospatialCollection}?x={x}&y={y}
GET: retrieves a coverage collection cell value array by CRS point. PUT: sets a coverage collection cell value array. DELETE: sets the coverage collection cell to NODATA. *The values of all bands of a Landsat multiband image corresponding to the x.y position are…*	
Tables	
Tables and query results	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}
GET: retrieves the list of record sets (tables and predefined queries). POST: adds a new table to a database *World population database by country, world illiteracy by country…*	
Record set (table, query result)	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}/{RecordSet}
GET: retrieves a record set. POST: adds a new record to a record set. PUT: replaces a table. DELETE: removes a record set *Word population database by country*	
Queries and views	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}/q={query}
POST: creates a new record set as a result of a query (e.g. SQL) and returns the record set name. *World countries with illiteracy greater than 50%*	
Field types	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}/{RecordSet}/_type
GET: retrieves the data model of this record set. PUT: updates the data model of this record set	

Resource	**URI template**
Supported methods and their meanings. ***Example***	
country name, string; illiteracy, percentage	
Record	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}/{RecordSet}/{record}
GET: retrieves a record (table row). PUT: replaces a record. DELETE: removes a record *NowhereLand, 20% illiteracy*	
Cell	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}/{RecordSet}/{record}/{field}
GET: retrieves a cell value. PUT: replaces a cell value. DELETE: blanks a cell value. *20% illiteracy*	
Cell links	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}/{RecordSet}/{record}/{field}/_link
GET: retrieves a list of links to cell values. POST: adds a new link. Note that this acts as an external data value that could be in the linked data semantic web. *www.countries.ccc/nowhereland, www.world.ccc/nowhereland*	
Cell link	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}/{RecordSet}/{record}/{field}/_link/{link}
GET: retrieves a particular link. PUT: modifies a link. DELETE: deletes a link.	
Field type	http://{OrganizationServer}/WWH/{SoftwareInstance}/{Database}/{RecordSet}/_all/{field}/_type
GET: retrieves a field type description. PUT: changes a field type. DELETE: removes a field *illiteracy, percentage (especially useful for updating one field keeping the others unchanged)*	

ANNEX III ACRÒNIMS / ACRONYMS

Aquesta és la llista d'acrònims que apareixen en la introducció, el resum de resultats i les conclusions d'aquesta tesi, i el seu significat. En general s'ha respectat la versió anglesa dels acrònims i la seva descripció original excepte en algun cas on hi ha una forta tradició a fer servir l'acrònim traduït. En aquesta tesi fem servir l'acrònim tant en singular com en plural sense afegir cap signe de plural ni duplicar lletres. El context permet saber si es parla en singular o en plural.

AIP	Architecture and Interoperability Pilot
API	Application Program Interface
CGI:	Common Gateway Interface
CEN	European Committee for Standardisation
CPU	Central Processing Unit
CSDGM	Content Standard for Digital Geospatial Metadata
CSW	Catalogue Service Web
DNS	Domain Name Service
ebRIM	E-Business Registry Information Model
FGDC	Federal Geographic Data Committee
GCI:	GEOSS Common Infrastructure
GCT	Geospatial Collection Table
GEMET	General Multilingual Environmental Thesaurus
GEO	Group on Earth Observations
GEOSS:	Global Earth Observation System of Systems
GeoRSS	Geo Really Simple Syndication[40]
GeoTIFF	Geo Tagged Image File Format
GIS	Geographical Information System
GIT	GIS Instance Table
GML	Geography Markup Language
GMLJP2	GML in JPEG 2000 for Geographic Imagery
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IARIA	International Academy, Research and Industry Association
ISO	International Organization for Standardization
JCR	Journal Citation Reports
JPEG	Joint Photographic Experts Group[41]
JPIP	JPEG 2000 Interactive Protocol
KML	Keyhole Markup Language[42]

[40] Originalment RDF Site Summary

[41] Encara que aquest acrònim es refereixi a un grup de treball, actualment representa un format de dades desenvolupat pel mateix grup

IDE	Infraestructura de Dades Espacial[43]
IDEC	Infraestructura de Dades Espacial de Catalunya[44]
IIS	Internet Information Service
ISO	International Organization for Standardization
KVP	Key and Value Pair
MMS	MiraMon Map Service
MMZ	MiraMon Map Zipped
MrSID	Multi-Resolution Seamless Image Database
NLB	Network Load Balance
NMA	National Map Agencies
OGC:	Open Geospatial Consortium
O&M	Observations and Measurements
OSGeo	Open Source Geo
OWS	OGC Web Services
PNG	Portable Network Graphics
ROA	Resource oriented architecture
REST	REpresentational State Transfer
RPC	Remote Procedure Call
RS	Remote Sensing
SCI	Science Citation Index
SDI	Spatial Data Infrastructure
SE	Symbology Encoding
SensorML	Sensor Markup Language
SHP	SHaPe file
SIF	Standards and Interoperability Forum
SIG	Sistema d'Informació Geogràfica[45]
SLD	Styled Layer Descriptor
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOS	Sensor Observation Service
SoS	System of Systems
SQL	Structured Query Language
SWE	Sensor Web Enablement
TC	Technical Committee

[42] Keyhole es el nom de l'empresa que inicialment va desenvolupar el format abans de ser comprada per Google. Actualment KML es un estàndard OGC i la versió 2.2 d'aquest estàndard ja no reconeix l'origen del nom d'aquest acrònim.
[43] The English translation of this acronym is SDI.
[44] Catalan SDI
[45] The English translation of this acronym is GIS.

TIFF	Tagged Image File Format
TMS	Tile Map Service
UUID	Universal and Unique Identifiers
URI	Universal Resource Identifier
URL	Universal Resource Location
VGI	Volunteered Geographic Information
W3C	World Wide Web Consortium
WCPS	Web Coverage Processing Service
WCS	Web Coverage Service
WFS	Web Feature Service
WFS-T	Web Feature Service Transactional
WKSS	Well Known Scale Set
WMC	OGC Web Map Context
WMS	Web Map Service
WMS-C	Web Mapping Service Cached
WMTS	Web Map Tile Service
WPS	Web Processing Service
WSDL	Web Service Definition Language
WWH	World Wide Hypermap
WWW	World Wide Web
XML	eXtensible Markup Language