OXYGENE : UNE PLATE-FORME OUVERTE POUR LE DÉPLOIEMENT DE SERVICES WEB GÉOGRAPHIQUES

Par Thierry BADARD, Arnaud BRAUN

Institut géographique national
Laboratoire COGIT, 2-4 avenue Pasteur, 94165 SAINT-MANDÉ Cedex, France
e-mail : {thierry.badard; arnaud.braun}@ign.fr

Résumé

Le développement d'applications web s'appuyant sur des données géographiques doit faire face à de nombreux problèmes : l'absence d'interopérabilité entre les modèles de données géographiques implémentés dans les SIG du commerce, l'utilisation de langages de programmation propriétaires, les faibles capacités des SIG commerciaux pour le déploiement d'applications sur Internet, etc. Afin de surmonter ces problèmes, de nombreuses technologies en génie logiciel ont émergé : des langages de programmation orientés-objet munis de capacités de communication sur les réseaux comme Internet, des techniques de modélisation basées sur la notion de composant logiciel, des systèmes de gestion de base de données objet-relationnel permettant le stockage et l'interrogation de données géographiques. Essayant de tirer profit de ces outils de plus en plus matures, une nouvelle plate-forme interopérable, baptisée OXYGENE, a été conçue et développée au laboratoire COGIT de l'IGN. Elle vise à offrir aux utilisateurs un cadre ouvert pour le développement de leurs applications et à permettre la centralisation des codes, leur documentation et une maintenance aisée de ceux-ci (pérennisation des travaux de recherche). En embarquant en son sein des protocoles et des langages standards pour l'appel de procédures à distance et la description de services, cette plate-forme vise à déployer les résultats des recherches entreprises au COGIT sous la forme de véritables services web. L'architecture et le modèle de la plate-forme OXYGENE seront tout d'abord présentés et détaillés. Le développement de services web géographiques sera ensuite discuté et illustré avec un exemple provenant d'une application mise en œuvre au laboratoire COGIT.

1 INTRODUCTION

Le développement d'applications web s'appuyant sur des données géographiques doit encore actuellement faire face à de nombreux problèmes :

- Le principal frein vient du manque d'interopérabilité entre les différents modèles de données géographiques mis en œuvre dans les logiciels SIG commerciaux, même si des efforts conséquents sont entrepris par l'ISO et l'OpenGIS pour les unifier et standardiser cette représentation commune. Une application développée sur un modèle non standard peut ainsi ne pas être réutilisable sur un autre système, sans y opérer des modifications conséquentes.
- Un autre problème majeur est que les langages de programmation liés aux logiciels SIG du marché sont très souvent des langages propriétaires. Outre le fait qu'ils nécessitent un long apprentissage afin d'en tirer parti, ils empêchent tout partage de code entre systèmes différents. On est alors obligé de coder plusieurs fois le même algorithme afin qu'il puisse fonctionner sous différentes plates-formes. De plus, les utilisateurs sont fortement dépendants des évolutions technologiques que peut entreprendre l'éditeur du logiciel SIG. La pérennité des développements n'est donc pas nécessairement assurée.
- Une autre entrave au développement d'applications géographiques sur le web est que la plupart des logiciels SIG commerciaux ne sont pas conçus nativement pour mettre en œuvre de telles solutions, sans acheter des modules additionnels relativement onéreux. De plus, lorsqu'ils sont dotés de fonctionnalités Internet, celles-ci se limitent à la mise à disposition de données géographiques (ex. : interfaces de consultation de données en ligne), mais ne permettent pas la distribution de traitements ou de processus (possibilité d'appeler à distance des algorithmes hébergés sur des serveurs - notion de service web -, évitant ainsi aux développeurs d'avoir à les coder ou d'être dans l'impossibilité de réaliser leur application, le traitement faisant appel à des techniques qu'il ne maîtrise pas forcément).
- Enfin, le fait que les SIG ne soient pas de véritables SGBD (Systèmes de gestion de bases de données) constitue une difficulté supplémentaire. Les problèmes de concurrence d'accès (possibilités de répartition de charge et de clustering) ou de gestion de la sécurité ne sont bien souvent pas adressés par les logiciels SIG du marché. Déployer des applications Internet sur de tels systèmes peut ainsi entraîner de nombreux problèmes d'efficacité, de rapidité et de cohérence.

Le laboratoire COGIT n'a pas échappé à ces problèmes. En effet, au cours des dix dernières années, 3 plates-formes de développement ont été mises au point en interne, afin de permettre aux chercheurs de répondre à leurs besoins de recherche :

- PlaGe (Plate-forme de généralisation) (1) : entièrement développée en ADA, elle a été conçue pour mettre au point les processus de généralisation cartographique peu contextuels (routes et bâti isolés).

  StratèGe (Stratégies pour la généralisation) : bâtie au-

dessus du moteur de Système expert SMECI vendu à l'époque par ILOG et développée en Le-Lisp (un Lisp propriétaire), elle a permis de coder les recherches en généralisation contextuelle (2). Les concepts et les algorithmes de StratèGe et PlaGe se retrouvent dans le prototype AGENT, un module consacré à l'automatisation du processus de généralisation à l'aide de technologies multi-agents. AGENT est une surcouche de la plate-forme SIG LaserScan™ (3).

- GéO2 (4) : Système de gestion de bases de données géographiques, orienté-objet, développé au dessus du SGBD O2 (O2 Technology, à l'époque également car n'est plus commercialisé et maintenu), principalement en O2C (un C/C++ propriétaire) qui a abrité et abrite encore, les recherches en gestion de l'information géographique (appariement, mise à jour, 3e dimension, multi-échelles, gestion du temps, ...).

De plus, certains développements (travaux d'étudiants pendant leur stage, certaines parties de thèses) ont été implémentés dans des plates-formes SIG du commerce (principalement ArcView™).

Cette diversité de systèmes et de langages ne pouvant pas communiquer entre eux a impliqué bien souvent de développer le même algorithme dans les différentes plates-formes. Les codes n'étant pas, pour la plupart, centralisés ou maintenus et documentés, ces mêmes traitements se sont vus implémentés plusieurs fois, au fil des personnes qui se sont succédé au laboratoire. Les systèmes utilisés en production ou à l'extérieur de l'IGN, étant pour la plupart des logiciels SIG commerciaux, le reversement des codes de recherche était rendu très difficile, voire impossible (modèles de données très divergents et temps d'apprentissage des langages trop longs). De plus, ces prototypes ont fait face à de gros problèmes de charge (volume de données gérable et rapidité des traitements), rendant ainsi peu rentable le déploiement tel quel de ces systèmes.

Afin de surmonter ces problèmes, de nombreuses technologies notamment en génie logiciel ont récemment émergé : des langages de programmation standardisés et extensibles, orientés objet, munis de capacités de communication sur les réseaux comme Internet (ex. : Java™), des techniques de modélisation basées sur la notion de composants logiciels (ex. : UML - Unified Modelling Language), des SGBD objet-relationnel permettant le stockage et l'interrogation de données géographiques (ex. : Oracle™ , Informix™ , PostGIS), etc.

Essayant de tirer profit de ces outils de plus en plus matures, le laboratoire COGIT a entrepris de concevoir et mettre en œuvre une nouvelle plate-forme interopérable de développement, visant à centraliser et pérenniser les travaux de recherche ainsi qu'à faciliter le reversement éventuel d'une partie des résultats auprès des unités de production de l'IGN. Cette plate-forme a été baptisée OXYGENE. A l'aide des technologies SOAP (Simple Object Access Protocol) (5) et WSDL (Web Service Description Language) (6), cette plate-forme a pour objectif de déployer les résultats de recherches entreprises au COGIT comme de véritables " services web géographiques ".

L'architecture et le modèle de la plate-forme OXYGENE seront tout d'abord présentés et détaillés (section 2). Le développement de services web géographiques sera ensuite discuté et illustré avec un exemple provenant d'une application mise en œuvre au laboratoire COGIT (section 3). Enfin, des perspectives relatives aux travaux futurs liés au développement de cette plate-forme seront fournies (section 4).

2 L'ARCHITECTURE D'OXYGENE

La structure générale de la plate-forme est illustrée sur la figure 1. OXYGENE. Elle repose sur deux principes :

- Elle est articulée autour du réseau pour permettre le déploiement des développements et la communication entre les composants.
- Les composants logiciels sont utilisés selon l'usage pour lequel ils sont le mieux adaptés.

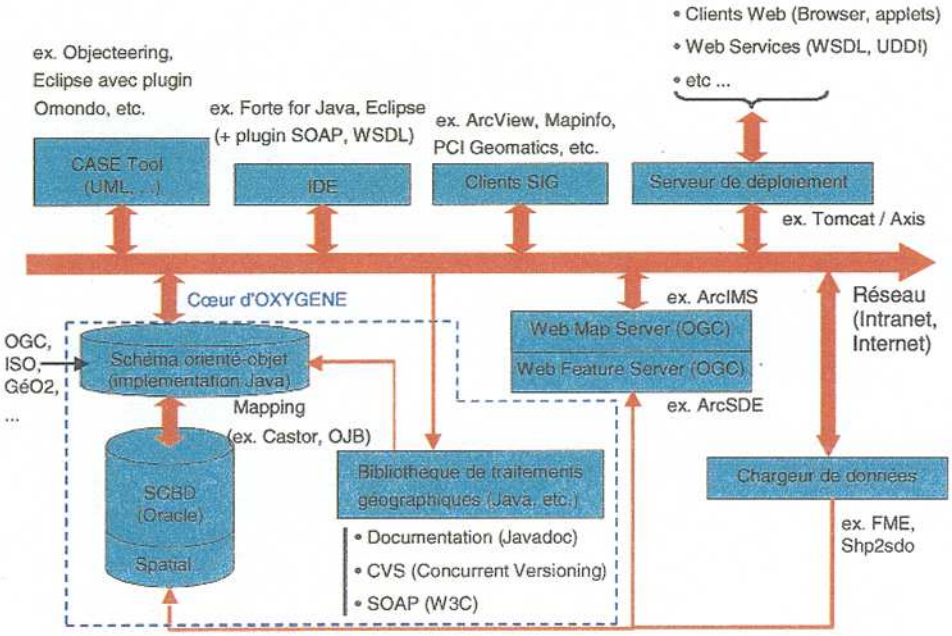

Figure 1 – Structure générale de la plate-forme OXYGENE

L'architecture s'appuie sur Java™ et sur des technologies open-source, et fournit aux utilisateurs un modèle de données objet, extensible, permettant de manipuler les objets géographiques, la géométrie, la topologie et les métadonnées, conformément aux spécifications de l'ISO et de l'OpenGIS (famille des standards 191xx). Les données sont stockées dans un SGBD relationnel (SGBDR) afin d'assurer un accès rapide et fiable au système, mais les utilisateurs n'ont pas à manipuler directement d'instructions SQL : ils modélisent leur application avec UML et codent en Java. Pour les utilisateurs, tout est objet ! La mise en correspondance (dite "mapping") entre les environnements objet et relationnel est automatique et flexible, grâce à des composants logiciels open source (cf. section 2.4). Pour assurer l'indépendance des développements, les opérateurs géographiques sont codés et stockés dans une bibliothèque indépendante (cf. section 2.2). Ces opérateurs proviennent de différentes sources : Internet, ou des développements antérieurs du laboratoire COGIT. Ils peuvent être écrits dans différents langages (C, C++, Fortran, Ada) et interfacés avec le langage Java. L'architecture d'OXYGENE inclut aussi différents outils, pour la génération automatique de documentation à partir du code, pour le partage de code (CVS – Concurrent Versioning System), pour le chargement de données géographiques, pour la modélisation (AGL – Atelier de génie logiciel), et pour le développement des applications (IDE – Environnement de développement intégré), et des clients SIG pour la visualisation et l'analyse des données.

2.1 Le schéma orienté-objet et son implémentation

Un schéma orienté-objet a été spécifié pour permettre la modélisation de tous les aspects de l'information géographique. Son organisation générale (paquetages UML) est illustrée sur la figure 2. Ce modèle implémente des standards de l'ISO (groupe de travail TC211) et des spécifications de l'OGC. Certains de ces standards sont finalisés, d'autres ne sont encore que des documents de travail :

- ISO 19107 - Spatial schema (7). Ce schéma permet la représentation de la géométrie et de la topologie. C'est un ensemble de classes regroupées dans un paquetage nommé *spatial*. Deux classes abstraites nommées *TP_Object* et *GM_Object* sont les classes mères pour modéliser respectivement la topologie et la géométrie. Les classes relatives à la topologie et à la géométrie sont structurées de manière parallèle, mais sont indépendantes. Ainsi, il est possible d'utiliser la topologie sans la géométrie : la topologie est vue comme une couche d'abstraction supérieure à la géométrie. Par exemple, cela permet la navigation dans un graphe routier (topologique) sans se préoccuper de sa représentation géométrique. Conformément au standard ISO, un *GM_Object* est une primitive (*GM_Primitive*) ou une collection de *GM_Object*. Une collection sans intersection ou recouvrement est un *GM_Aggregate*. Une collection d'objets géographiques connectés géométriquement est un *GM_Complex*. La classe *GM_Complex* permet ainsi de représenter le partage de primitives géométriques entre différents objets géographiques, ou la modélisation de structures telles qu'une séquence de courbes connectées (par exemple pour un réseau routier).
- ISO 19109 - Rules for application schema (8). Ce schéma permet la définition du métamodèle d'une application géographique, c'est-à-dire le métamodèle pour les classes géographiques que cette application manipule. Ce standard a été mis en œuvre via l'implémentation d'un certain nombre de classes, regroupées dans le paquetage *dico*. Ces classes permettent donc de constituer le dictionnaire (i.e. le type des attributs, l'ensemble de ses valeurs possibles, etc.) des données modélisées et stockées dans la plate-forme.
- ISO 19111 - Spatial referencing by coordinates (9). Cette norme traite des systèmes de coordonnées et de projections. À l'heure actuelle, comme aucun changement de système de coordonnées ou de projection ne nous était nécessaire, les jeux de données utilisés au laboratoire COGIT étant tous en Lambert II étendu, ce standard n'a pas été mis en œuvre dans OXYGENE. Seule une classe abstraite dans le paquetage *src* (pour «Spatial Referencing by Coordinates») a été développée afin de conserver la conformité au standard ISO.
- ISO 19115 - Metadata (10). Seulement une partie du modèle complet de métadonnées défini par l'ISO a été implémentée. Ses classes sont regroupées dans le paquetage *metadata*. Seules les plus pertinentes et correspondant à ce que l'on peut communément trouver à l'IGN ou en dehors de l'Institut, ont été codées. Une implémentation complète de cette norme est probable, notamment afin de satisfaire les besoins en terme de modélisation de métadonnées, des actions de recherche en cours au laboratoire sur la consultation et l'unification des bases de données géographiques.

Un paquetage additionnel (appelé *geoschema*) a été conçu pour rassembler toutes les classes d'objets géographiques issues du chargement de jeux de données ou créées et stockées dans le système. Ce paquetage contient notamment une classe (nommée *FT_Feature*) dont toutes les classes géographiques héritent. *FT_Feature* joue donc le rôle de classe mère pour l'ensemble des objets géographiques. Cette classe porte la topologie et la géométrie. Un utilisateur définira ainsi un schéma d'application adapté à ses données géographiques dans le paquetage *geoschema*, en utilisant le métamodèle du paquetage *dico*.

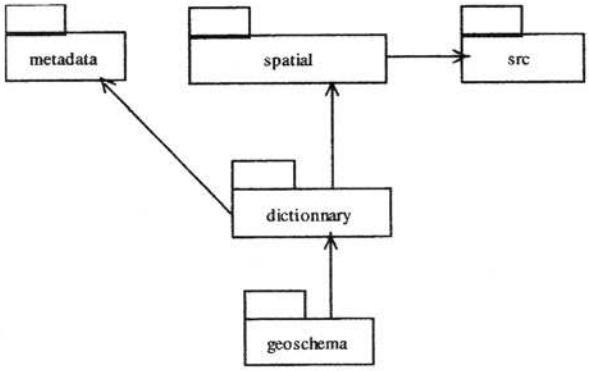

Figure 2 – Organisation générale du modèle (diagramme de paquetages UML)

Deux outils complémentaires ont été développés pour bénéficier pleinement de ce point de vue orienté-objet sur les données géographiques : un visualisateur d'objets géographiques, basé sur le projet open-source Geotools (11), et un butineur graphique d'objets conçu pour afficher les différents états des objets et appeler de manière interactive les méthodes.

Les chercheurs du laboratoire COGIT ont commencé à implémenter des applications sur la plate-forme OXYGENE. Ces applications concernent les spécifications des bases de données, les études de risques, l'appariement de données géographiques, la propagation automatique des mises à jour, l'enrichissement de MNT avec des données vecteur (12). Afin de satisfaire à ces exigences de recherche, les utilisateurs créent leur propre modèle de données. Comme illustré sur la figure 3, chaque schéma applicatif s'appuie sur le cœur du schéma d'OXYGENE directement ou à travers le modèle des "cartes topologiques". Ceci fournit aux utilisateurs un moyen de modéliser leurs données géographiques de manière conforme au concept de carte topologique modélisé dans (13).

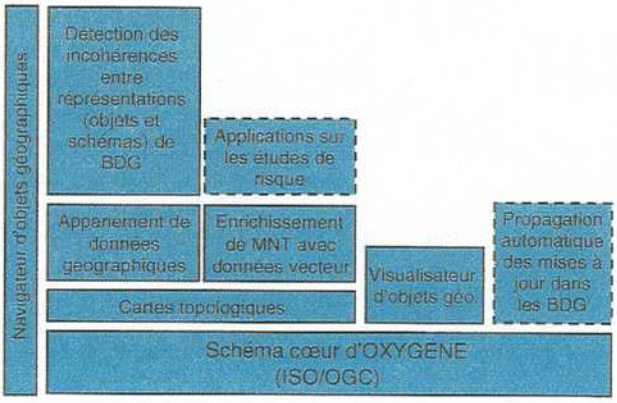

Figure 3 – Développement d'applications avec OXYGENE

2.2 Bibliothèque d'opérateurs géographiques

Bien que s'appuyant sur la structure du modèle de la plate-forme, les opérateurs et traitements géographiques sont développés dans des bibliothèques séparées et partagées afin de préserver l'indépendance des développements vis-à-vis des changements que pourrait subir OXYGENE. Ces bibliothèques regroupent ainsi non seulement des algorithmes de base sur les primitives géométriques (union, intersection, buffer, etc.) mais aussi des traitements plus complexes comme des algorithmes de triangulation de Delaunay, de calcul de diagramme de Voronoï ou de mesure de la distance de Hausdorff. Ces algorithmes sont majoritairement développés en Java™.

Bien que les développements menés jusqu'alors dans les différentes plates-formes du laboratoire COGIT (PlaGe, Stratège, GéO2) aient été écrits dans des langages de programmation autres que Java, leur intégration au sein d'OXYGENE demeure possible et ceci sans avoir à recoder entièrement tous les algorithmes. En effet, l'interface JNI (Java Native Interface) fournie par Sun Microsystems (14) rend possible une telle réutilisation de programmes compilés dans des langages tiers. Un simple fichier d'interface en Java est à écrire afin de spécifier comment s'opère la conversion des types de données entre Java et ceux utilisées par exemple dans une bibliothèque compilée C/C++, ADA ou Fortran. JNI évite ainsi de devoir à nouveau développer des codes qui ont déjà nécessité d'importants efforts pour leur stabilisation et leur optimisation (anciens développements, bibliothèques open source de traitements géométriques disponibles sur l'Internet, etc.). De surcroît, bien que Sun Microsystems fournisse, de version en version, des machines virtuelles Java permettant de développer des applications de plus en plus rapides, l'intégration de tels programmes au sein de la plate-forme permet globalement d'obtenir des traitements présentant des temps d'exécution plus courts que s'ils avaient été écrits entièrement en Java.

2.3 Le système de gestion de base de données (SGBD)

Les logiciels SIG permettent le stockage et la manipulation de données géographiques, mais ces solutions n'offrent pas l'ensemble des caractéristiques et fonctionnalités que peuvent offrir les SGBD. Les problèmes de concurrence d'accès, de gestion de la sécurité ou encore le stockage de volumes de données conséquents ne sont, en effet, bien souvent pas gérés par les logiciels SIG du marché. Comme l'un des objectifs fixés dès la phase de conception d'OXYGENE était d'utiliser un composant pour ce à quoi il était destiné et répondant bien à ce but, une solution basée sur un SGBD a été retenue. Celui-ci est relationnel car les SGBD orienté-objet ont montré leurs limites en terme de performance (mauvaise tenue à la montée en charge, temps prohibitifs de réponse aux requêtes). Le SGBD Oracle a été retenu pour OXYGENE. Oracle est muni d'une extension spatiale qui lui permet de manipuler l'information géométrique, sous une forme «objet» depuis la version 8i (15). Par ailleurs, la structure modulaire de la plate-forme permettrait d'utiliser tout autre SGBD capable de stocker de l'information géographique, comme par exemple PostGIS, l'extension spatiale open-source du SGBD PostgreSQL (16).

2.4 Le lien objet-relationnel

Oracle est un SGBD relationnel qui possède certes quelques fonctionnalités «objet», mais il reste fondamentalement relationnel. Le modèle de la plate-forme a été implé-

menté en Java et définit un point de vue entièrement orienté-objet sur les données stockées dans le système. Un lien entre ces deux «mondes» devait donc être établi : c'est ce que permet le processus de «mapping» (une table stockée dans le SGBD relationnel correspond à une classe Java dans le modèle). Pour ce faire, deux API Java open-source ont été testées et mises en œuvre : CASTOR (17) et OJB (18). A l'heure actuelle, OJB est plus rapide que CASTOR et offre des fonctionnalités de «mapping» objet/relationnel plus avancées. Ces deux solutions s'appuient sur la norme JDO spécifiée par Sun™ (19) et permettent le stockage flexible des informations de correspondance dans des fichiers XML, humainement lisibles, et ainsi facilement modifiables pour réaliser un mapping performant (i.e. adapté aux besoins de l'application). Grâce à ce «mapping» transparent, les utilisateurs de la plate-forme ne manipulent pas directement les tables Oracle et peuvent continuer à raisonner et développer entièrement en orienté-objet. La structure modulaire d'OXYGENE permet de surcroît de supporter tout autre outil de «mapping», simplement en implémentant une nouvelle classe Java, dérivée des interfaces dédiées à la connexion aux systèmes de gestion de base de données, fournies avec la plate-forme.

2.5 Les outils de développement

Comme pour n'importe quel projet de développement, les codes d'OXYGENE sont documentés. À partir de commentaires insérés dans le code, un outil (Javadoc™ , fourni avec le kit de développement Java SDK) génère automatiquement une documentation sous la forme de pages HTML. La cohérence entre les développements et la documentation est ainsi assurée. La documentation peut être consultée sur un serveur web interne par les développeurs. D'autres documents électroniques (cours, transparents, articles, livres) sont aussi disponibles sur ce serveur pour faciliter l'apprentissage des différentes techniques et technologies mises en œuvre dans OXYGENE.

Le code est partagé entre les différents chercheurs grâce à un CVS (Concurrent Versionning System) (20). Cet outil centralise les codes sur un serveur. Chaque utilisateur peut y charger les mises à jour et y écrire ses propres modifications du code. Les conflits éventuels (ex. : modification d'une même portion de code par différents utilisateurs) sont signalés et des outils sont à disposition pour les résoudre et ainsi assurer le maintien de la cohérence. L'historique est également conservé. L'accès au CVS se fait sur un serveur accessible en interne à l'IGN.

La conception d'une application passe tout d'abord par sa modélisation. Cette étape peut être facilitée par des outils de génie logiciel (type AGL ou outils CASE), qui permettent de saisir des schémas conceptuels, de les documenter et de les transformer éventuellement en schémas logiques en fonction des SGBD ou des langages de développement cibles. Il existe de nombreux outils tant sur le marché qu'en open-source permettant d'atteindre ce but. Ils s'appuient sur différents formalismes. En raison de tous les concepts utiles et intéressants (différents points de vue sur une application, paradigme orienté-objet, etc.) qu'il fournit, UML a été adopté en tant que formalisme commun pour la conception des différents schémas d'application mis au point par les chercheurs du laboratoire. Des formalismes plus spécifiques comme Perceptory (21) ou MADS (22), qui permettent la modélisation directe de l'information géographique et temporelle, pourraient être utilisés dans OXYGENE.

Afin de faciliter le développement des applications, OXYGENE est muni d'un IDE (Environnement de développement intégré). Ce sont des logiciels bien connus des programmeurs qui permettent d'éditer du code, de le compiler, de l'exécuter, de le déboguer, de gérer le partage de code en tant qu'utilisateur CVS, de visualiser le contenu des tables du SGBD, d'exécuter des requêtes dans le SGBD. L'IDE principalement déployé au laboratoire COGIT est Eclipse (23). Il s'agit d'un projet open source, initialement basé sur une contribution d'IBM. Comme les sources sont accessibles, de nombreux «plugins» ont été développés par la communauté open-source et sont disponibles (souvent gratuitement) sur l'Internet. Ils fournissent à Eclipse des fonctionnalités de plus à plus avancées, le transformant ainsi en véritable AGL : développement et déploiement de services web, modélisation UML (24), connexion aux systèmes de gestion de bases de données, etc.

2.6 Chargement et visualisation des données géographiques

Les données sont généralement fournies dans un des formats du marché (ESRI™ Shapefile, MapInfo™ MIF/MID, GeoConcept™ , etc.). Afin que les utilisateurs de la plate-forme puissent y accéder et les manipuler depuis leurs développements Java, les données doivent être chargées dans le SGBD Oracle. Pour cela, un logiciel appelé traducteur est utilisé. Il permet aussi d'exporter les données depuis Oracle vers un autre format, rendant les résultats de traitements disponibles auprès de n'importe quelle plate-forme SIG. Le logiciel utilisé est FME (Feature Manipulation Engine) (25).

La visualisation peut se faire à l'aide d'un simple «visualisateur» ou avec un des SIG classiques du marché (MapInfo, GeoConcept, Arcview) qui prévoient tous des moyens de se connecter à Oracle. On peut alors profiter de toutes les puissantes fonctions d'affichage et d'analyse de ces produits. Un simple visualisateur peut aussi être utilisé (celui de FME, ou Geomatica de PCI Geomatics par exemple). Un visualisateur d'objets géographiques a néanmoins été développé au laboratoire COGIT (cf. section 2.1), afin de pouvoir visualiser et accéder aux propriétés des objets dynamiquement (i.e. depuis et à l'intérieur d'un programme Java en cours d'exécution), entièrement en objet et non plus en relationnel, comme le permettent les outils précédents car accédant au SGBD directement. Ce visualisateur permet de surcroît la visualisation des objets non persistants, c'est-à-dire créés au cours de l'exécution d'un programme, de déclencher interactivement les méthodes des objets (persistants ou non) ou de visualiser leurs différents états dans le temps à l'aide d'un navigateur graphique d'objets Java générique, également en cours de développement au laboratoire. Ce visualisateur est basé principalement sur les outils open source développés par le projet Geotools (11).

2.7 Les services web géographiques

Comme mentionné en introduction, en embarquant en son sein dès sa conception, des technologies comme SOAP (Simple Object Access Protocol) (5) et WSDL (Web Services Description Language) (6), édictées par le W3C, cette plate-forme vise à déployer les résultats des recher-

ches entreprises au COGIT, sous la forme de véritables services web. A l'heure actuelle, de tels services s'appuyant sur des données géographiques ne sont pas encore entièrement standardisés même si des efforts conséquents sont entrepris par l'ISO et l'OpenGIS Consortium (OGC) pour tenter d'établir des spécifications permettant leur interopérabilité sémantique.

2.7.1 La taxinomie de l'OGC

L'OGC a proposé une architecture pour les services web (« The OpenGIS Service Architecture ») qui a été adoptée comme un standard ISO, ISO 19119 (26). Une taxinomie des services web géographiques s'appuyant sur le modèle OSE (Extended Open Service Environment) est proposée dans ce standard. Cinq grandes catégories de services ont ainsi été identifiées :

- Les services interactifs : visualisateurs et éditeurs.
- Les services de gestion de l'information : accès aux données persistantes.
- Les services de flux et de tâches : définition de chaînage de services.
- Les services de traitements : modification de la donnée géographique.
- Les services de communication : encodage et infrastructure.

2.7.2 Web Feature Server and Web Map Service

Les Web Map Service et Web Feature Server appartiennent respectivement à la première et à la seconde catégorie de la taxinomie de la section 2.7.1. Un «Web Feature Server» (WFS), tel que spécifié par l'OGC (27), permet de connecter des entrepôts de données géographiques pour les mettre à disposition des utilisateurs via l'Internet. Il devient alors possible de manipuler les données au travers du web : des opérations de création, de destruction et de mise à jour de données, ainsi que des requêtes de sélection sont offertes par un tel service. Un «Web Map Service» (WMS), tel que défini par l'OGC (28), produit une carte à partir de données géographiques dans un navigateur web, une carte étant ici une représentation visuelle des données et non les données elles-mêmes. Les cartes peuvent être dans un format image (ex. : GIF, JPEG) ou dans un format vecteur, comme SVG. Actuellement, ni WFS, ni WMS n'ont été mis en œuvre dans OXYGENE. Néanmoins, l'architecture modulaire de la plate-forme le permet. Des implémentations de WMS et WFS en open-source sont d'ores et déjà disponibles sur l'Internet, et leur intégration au sein de la plate-forme OXYGENE est planifiée.

2.7.3 Vers des services web de traitement

Un service web est un ensemble de méthodes qui peuvent être appelées à distance à travers un réseau, (comme l'intranet d'une entreprise, ou Internet lui-même). L'information dont doit disposer une application pour invoquer un service web dans un programme est donné par un document WSDL (Web Service Description Language). Les documents WSDL sont indexés dans des registres facilement consultables baptisés UDDI (Universal Description, Discovery, and Integration), afin que les développeurs et les applications puissent les localiser. Les services web sont rendus disponibles par des serveurs d'application acceptant le protocole SOAP (Simple Object Access Protocol) qui s'appuie sur le langage XML (eXtensible Markup Language) (29). L'ensemble des services web disponibles sur tout l'Internet est appelé le Service Web. Le Service Web sera l'épine dorsale de la prochaine génération d'applications distribuées. Aujourd'hui, le Service Web n'en est qu'à son enfance, mais sa croissance est très prometteuse.

Le but d'OXYGENE est de déployer les recherches entreprises au COGIT sous la forme de "services web de traitements" selon la taxinomie de l'OGC (cf. Section 2.7.1). De tels services existent déjà sur Internet : services de transformations de coordonnées ou services d'orthorectification par exemple. Nous avons testé l'utilisation des technologies SOAP et WSDL, car elles deviennent standards et matures pour le déploiement de services web non géographiques. Néanmoins, l'attention est gardée sur les recommandations de l'OGC pour les services web géographiques. Le paragraphe suivant montre un exemple de déploiement d'un service web, s'appuyant sur ces technologies.

3 EXEMPLE DE DÉPLOIEMENT ET D'UTILISATION D'UN SERVICE WEB DANS OXYGENE

L'exemple détaillé dans cette section illustre le déploiement d'un service web géographique et son utilisation par un client. Bien que le serveur et le client soient, pour des raisons pratiques, tous les deux mis en œuvre au sein de la plate-forme OXYGENE, la faisabilité d'un tel déploiement est prouvée. Des tests de déploiement de services web géographiques dans des environnements distribués et hétérogènes (OXYGENE en tant que serveur sur une machine Unix ou Linux et un logiciel SIG du marché en tant que client sur un système d'exploitation Windows, par exemple) sont prévus.

Comme mentionné en section 2.7.3, le service web détaillé dans cette section s'appuie sur une implémentation plutôt «traditionnelle» de tels composants. Le protocole SOAP a été ainsi mis en œuvre pour réaliser les échanges de messages entre le client et le serveur. Un serveur d'application composé de Tomcat et d'Apache SOAP a été utilisé pour déployer le service. Le standard WSDL permettant la description des services web n'est actuellement pas supporté par Apache SOAP. Une migration vers Apache Axis (30), la nouvelle implémentation de référence du protocole SOAP du W3C fournie par l'Apache Software Fundation le permettra bientôt. Des tests de cette nouvelle architecture logicielle de déploiement seront réalisés dans un futur proche.

3.1 Présentation du service web déployé

L'exemple de service web illustré dans cette section est nommé «FiltrageService». Il réalise le déploiement d'un algorithme simple de filtrage de type Douglas-Peucker, implémenté dans la plate-forme OXYGENE. Comme cet algorithme (méthode) est appliqué à un vecteur de points (nommé «DirectPosition» dans le modèle ISO de la plate-forme) envoyé par le client au serveur, on n'accède pas ici au SGBD. La figure 4 illustre l'architecture N-tiers sur laquelle le déploiement de ce service web s'appuie.

Figure 4 – Architecture N-tiers du serveur de déploiement d'OXYGENE

Le client (un programme Java) accède tout d'abord à un objet géographique stocké dans le SGBD par le biais de méthodes d'accès JDO. La méthode distante de filtrage, interfacée par le service « FiltrageService « déployé sur le serveur, est alors appliquée à l'objet. Le client envoie tous les paramètres au serveur sous la forme de messages SOAP. Le serveur traite la requête en déclenchant les méthodes correspondantes dans la bibliothèque de traitements disponibles dans OXYGENE et retourne le résultat également sous la forme de messages SOAP. Le client peut alors achever la transaction qu'il a initiée et la valider (en rendre les effets persistants). Comme mentionné précédemment, à la fois le client et le serveur sont mis en œuvre au sein de la plate-forme. La figure 5 permet d'identifier clairement les composants de son architecture utilisés soit par le serveur, soit par le client.

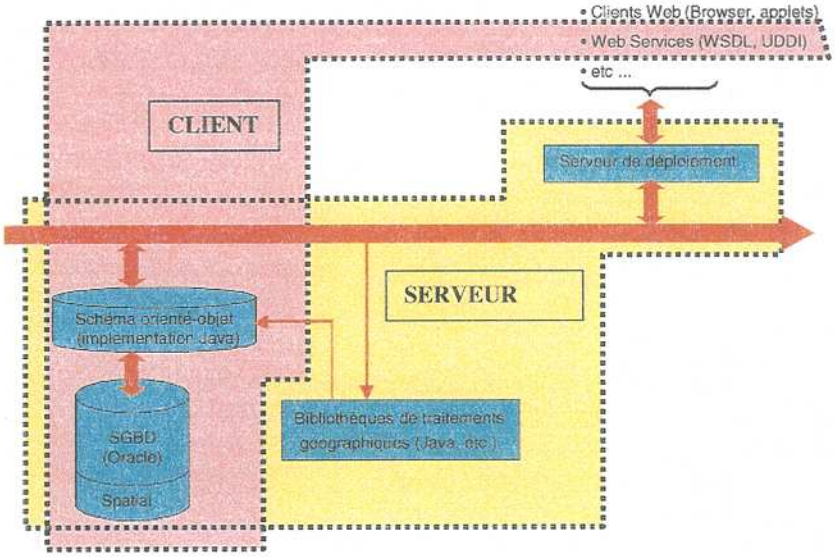

Figure 5 – L'exemple dans l'architecture globale

Les sections suivantes fournissent les détails de la mise en œuvre de cet exemple de déploiement de service web tant du côté du client que du côté serveur. Les messages SOAP échangés sont également présentés.

3.2 Le serveur

Sur le serveur, une interface (nommée «FiltrageService») est tout d'abord implémentée. Elle porte un certain nombre de méthodes relatives au traitement de filtrage. L'une de ces méthodes correspond au filtrage de type Douglas-Peucker pour un vecteur de points. Elle encapsule simplement un appel vers la méthode de filtrage correspondante, disponible dans la bibliothèque de traitements partagés de la plate-forme. Le déploiement consiste alors simplement en l'écriture d'un fichier XML «descripteur du déploiement du service web» (WSDD, Web Service Deployment Descriptor), nécessaire au serveur de déploiement afin qu'il puisse répondre aux requêtes initiées par les clients. Cette réponse est assurée par le serveur de déploiement Apache SOAP 2.2 qui tire bénéfice du moteur de servlet Tomcat et implémente ses propres servlet afin de récupérer les requêtes des clients, invoquer les méthodes demandées (définies dans le fichier WSDD) et envoyer les réponses aux clients.

3.3 Le client

Les objets géographiques sur lesquels ce test de déploiement est réalisé, sont des tronçons de route provenant de la BDCarto®, une base de données produite par l'IGN. Sur le client, les données ont été au préalable chargées dans le SGBD Oracle. Elles sont ainsi stockées sous la forme de tables relationnelles. Un outil fourni avec la plate-forme OXYGENE génère automatiquement les classes Java correspondantes et un fichier de mapping XML, que l'outil de mapping objet-relationnel peut interpréter (cf. figure 6). Les classes Java générées héritent de la classe *FT_Feature* qui porte la géométrie et la topologie. Ces dernières peuvent ainsi être accédées directement depuis la classe générée.

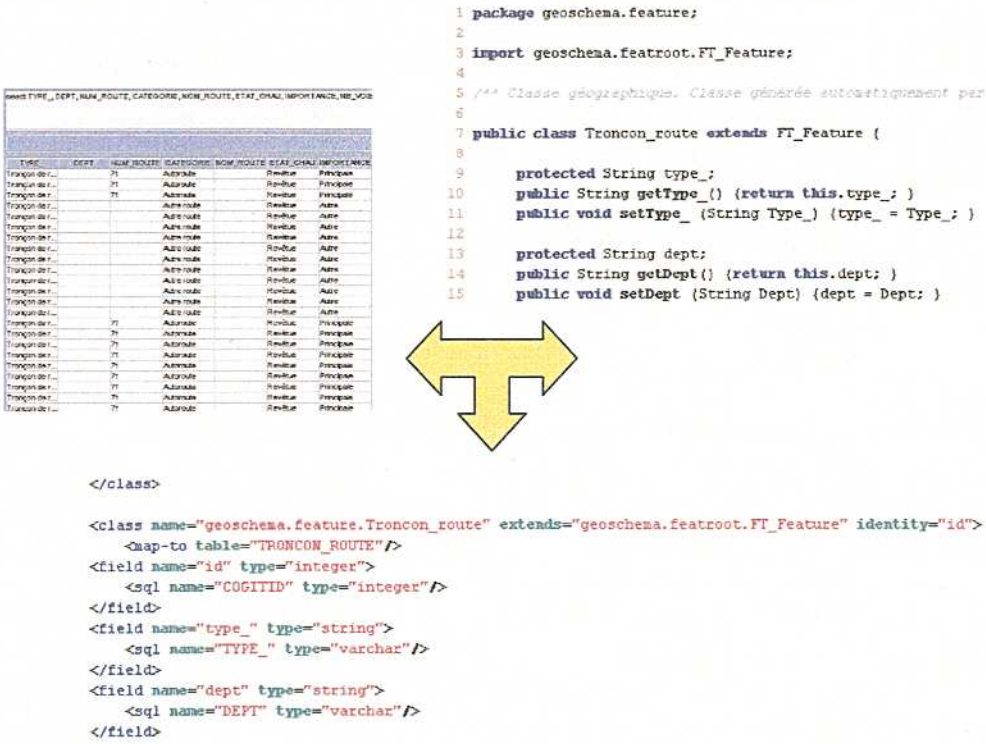

Figure 6 – Table du SGBD, classe java et fichier XML de mapping

L'extrait de code suivant (cf. figure 7) illustre comment est réalisé l'appel au traitement distant de filtrage de Douglas-Peucker, déployé sur le serveur.

```
// Initialise a new database connection
Geodatabase db = new GeodatabaseOjbOracle();

// Initialise a Java stub with an URL
SoapClientHelper.initService(FiltrageService.class, DeploymentDesciptor.xml,
  "http://localhost:8080/soap/servlet/rpcrouter");
FiltrageService filtre = (FiltrageService)
  SoapClientHelper.makeServiceInstance(FiltrageService.class);

// Start a transaction and load a feature by its class and identifier
db.begin();
FT_Feature feature = (FT_Feature)db.load(roadSectionClass,gid);

// Get its geometry and apply the remote Douglas-Peucker on it
GM_LineString linestring1 = (GM_LineString)feature.getGeom();
GM_LineString linestring2 = new GM_LineString
  (filtre.DouglasPeuckerList(linestring1.coord(), threshold));

// Create a new feature and set the new geometry
FT_Feature result = new Result_Curve();
Result.setGeom(linestring2);

// This feature is made persistent in the database and transaction is committed
db.makePersistent(result);
db.commit();
```

Figure 7 – Exemple de code Java réalisant un appel à un service distant via SOAP

Une instance de la classe Geodatabase est tout d'abord créée. Elle permet l'ouverture d'une connexion vers la base de données géographiques stockées dans le SGBD Oracle au travers du logiciel de mapping objet-relationnel OJB. Un «stub» (ou «proxy») Java est alors initialisé. Il permet l'appel dynamique des méthodes distantes portées par l'interface «FiltrageService» déployée sur le serveur. Son implémentation est réalisée en fournissant l'URL du fichier de description de déploiement du service web (WSDD) auquel on souhaite accéder et l'URL de la servlet d'écoute («listener») SOAP à un «SOAP client helper». Une transaction est alors initiée et un objet persistant est chargé en fournissant le nom de la classe et l'identifiant de l'objet auquel on souhaite accéder. La géométrie de l'objet géographique (un tronçon de route) est alors accédée et le traitement de Douglas-Peucker distant est appliqué sur

celle-ci. Un nouvel objet géographique est créé et la géométrie résultant du processus distant est affectée à cet objet. Il est enfin stocké dans la base et la transaction est validée.

3.4 Échange de message SOAP

Comme mentionné dans la section 3.1, les échanges d'informations entre le serveur et le client sont basés sur le protocole SOAP (Simple Object Access Protocol). Ce dernier s'appuie sur le protocole HTTP (Hypertext Transfert Protocol) du World Wide Web et XML (Extensible Markup Language). SOAP spécifie comment encoder une entête HTTP et un fichier XML afin qu'un programme exécuté sur un ordinateur avec un système d'exploitation particulier puisse appeler un service sur une autre machine avec un système d'exploitation éventuellement différent. SOAP spécifie également comment ces informations sont transférées et comment la méthode distante retourne son résultat (réponse). Les messages SOAP sont automatiquement générés et interprétés à la fois par le client (au travers de son «proxy») et par le serveur de déploiement. Ainsi, un programmeur n'a à aucun moment à s'en soucier.

Un outil fourni avec le serveur de service web Apache SOAP 2.2 permet de «suivre» les messages échangés. La figure 8 fournit un exemple de message SOAP échangé entre le client et le serveur lorsque la méthode de filtrage de Douglas-Peucker distante est exécutée sur le serveur. On peut ainsi clairement voir apparaître les coordonnées de l'objet géométrique sur lequel ce traitement s'applique (un tronçon de route).

```
<CRS xsi:type="xsd:int">0</CRS>
</item>
<item xsi:type="ns1:directposition">
<x xsi:type="xsd:double">1000512.57</x>
<dimension xsi:type="xsd:int">3</dimension>
<z xsi:type="xsd:double">280.3</z>
<coordinate
xmlns:ns24="http://schemas.xmlsoap.org/soap/encoding/"
    xsi:type="ns24:Array"ns24:arrayType="xsd:double[3]">
<item xsi:type="xsd:double">1000512.57</item>
<item xsi:type="xsd:double">1870538.61</item>
<item xsi:type="xsd:double">280.3</item>
</coordinate>
<y xsi:type="xsd:double">1870538.61</y>
<CRS xsi:type="xsd:int">0</CRS>
</item>
<item xsi:type="ns1:directposition">
<x xsi:type="xsd:double">1000511.78</x>
<dimension xsi:type="xsd:int">3</dimension>
<z xsi:type="xsd:double">281.5</z>
<coordinate
xmlns:ns25="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns25:Array"
    ns25:arrayType="xsd:double[3]">
<item xsi:type="xsd:double">1000511.78</item>
<item xsi:type="xsd:double">1870522.71</item>
<item xsi:type="xsd:double">281.5</item>
</coordinate>
<y xsi:type="xsd:double">1870522.71</y>
<CRS xsi:type="xsd:int">0</CRS>
</item>
</list>
</dummy>
<dummy xsi:type="xsd:int">5</dummy>
</ns1:DouglasPeuckerList>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 8 – Exemple d'échange de message SOAP entre le client et le serveur

4 CONCLUSIONS ET PERSPECTIVES

OXYGENE, la nouvelle plate-forme interopérable conçue et développée au laboratoire COGIT de l'IGN a été présentée et détaillée. Elle offre aux utilisateurs un cadre de travail ouvert pour le développement d'applications basées sur les données géographiques, et permet la centralisation des codes, leur documentation et donc des facilités pour leur maintenance et leur mise à jour (préservation des développements). Des chercheurs du COGIT ont commencé à développer des applications à l'aide de cette plate-forme. Ces applications concernent les spécifications des bases de données géographiques, les études de risques, l'appariement de données géographiques, la propagation automatique des mises à jour, l'enrichissement de MNT avec des données vecteur, etc.

En embarquant des technologies comme des protocoles standards pour l'appel distant de procédures, et des langages pour la description de services, la plate-forme OXYGENE permet le déploiement des recherches entreprises au COGIT comme de véritables "services web géographiques". La faisabilité d'un tel déploiement a été prouvée. Néanmoins, il reste un travail important pour réaliser des implémentations entièrement conformes aux standards de l'ISO et de l'OGC. Nos tâches futures concerneront principalement de telles implémentations.

REFERENCES

1. F. Lecordix, C. Plazanet, J.-P. Lagrange, *A Platform for Research in Generalisation: Application to Caricature,*. GeoInformatica 1:2, Kluwer Academic Publishers, Boston, pp.161-182 (1997)
2. A. Ruas, *OO-Constraint Modelling to Automate Urban Generalisation Process*, Proceedings of the 8th International Symposium on Spatial Data Handling (SDH'98) pp.225-235 (1998)
3. M. Barrault, N. Regnauld, C. Duchêne, K. Haire, C. Baeijs, Y. Demazeau, P. Hardy, W. Mackaness, A. Ruas, R. Weibel, *Integrating Multi-agent, Object-oriented, and Algorithmic techniques for Improved Automated Map Generalization,* Proceedings of the 20th International Cartographic Conference, vol. 3, Beijing, China, pp. 2110-2116 (2001)
4. L. Raynal, B. David, G. Shorter, *Building an OOGIS prototype: experiments with GeO2,* AutoCarto, pp.137-146 (1995)
5. World Wide Web Consortium, *Simple Object Access Protocol (SOAP) 1.1,* W3C Note, http://www.w3c.org/TR/SOAP (2000)
6. World Wide Web Consortium, *Web Service Description Language (WSDL) 1.1,* W3C Note, http://www.w3c.org/TR/WSDL (2001)
7. OpenGIS™ Consortium, *The OpenGIS™ Abstract Specification – Topic 1: Feature Geometry (ISO 19107 Spatial Schema),* version 5, OGC document number 01-101 (2001)
8. ISO, *Geographic Information, Rules for Application Schema,* ISO Draft International Standard 19109 (2001)
9. ISO, *Geographic Information, Spatial Referencing by Coordinates,* ISO Draft International Standard 19111 (2001)
10. ISO, *Geographic Information, Medatada,* ISO Draft International Standard 19115 (2001)
11. Geotools Project, *Geotools, Open Source mapping toolkit,* http://www.geotools.org (2003)
12. F. Rousseaux, *Enrichment of a TIN with 3D vectors data,* GIS Research in UK (GISRUK), City University, London (2003)
13. B. David, *Modélisation, représentation et gestion d'information géographique – Une approche en relationnel étendu,* PhD. Thesis, Université Paris 6 (1991)
14. Sun™ Microsystems, *Java Native Interface (JNI),* http://java.sun.com/products/jdk/1.2/docs/guide/jni (2003)
15. Oracle™, *Oracle Technology Network,* http://otn.oracle.com (2003)
16. PostGIS™, *Geographic Objects for PostgreSQL,* http://postgis.refractions.net (2003)
17. Castor Project, *Castor Project Home Page,* http://castor.exolab.org (2003)
18. The Apache DB Project, *OJB – ObJectRelationalBridge,* http://db.apache.org/ojb/ (2003)
19. Sun™ Microsystems, *Java Data Objects (JDO),* http://java.sun.com/products/jdo (2003)
20. CVS, *Concurrent Version System, the Open Standard for Version Control,* http://www.cvshome.org (2003)
21. Bédard, Y., *Visual Modelling of Spatial Databases: Towards Spatial PVL and UML,* Geomatica, Vol. 53, No. 2., pp. 169–186 (1999)
22. Parent, C., Spaccapietra, S., Zimanyi E., Spatio-Temporal Conceptual Models: Data Structures + Space + Time, Proceedings of the 7th ACM Symposium on Advances in GIS, Kansas City, Kansas (1999)
23. Eclipse Project, *Eclipse Main Page,* http://www.eclipse.org (2003)
24. Omondo, *Eclipse-Omondo, the Live UML Company,* http://www.omondo.com (2003)
25. Safe™ Software, *Data Translation Solutions,* http://www.safe.com (2003)
26. OpenGIS™ Consortium, *The OpenGIS™ Abstract Specification – Topic 12: Service Architecture (ISO 19119 Geographic Information Services),* version 4.3, OGC document number 02-112 (2002)
27. OpenGIS™ Consortium, *Web Feature Server Specification,* OGC document number 01-023 (2001)
28. OpenGIS™ Consortium, *Web Map Service Implementation Specification,* OGC document number 01-068r3 (2001)
29. A. Ryman, *Understanding Web Services,* http://www7.software.ibm.com/vad.nsf/Data/Document4362/ (2003)
30. Apache Axis, *Home Page, http://ws.apache.org/axis/* (2003)