CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

MEMOIRE

présenté par **Christine Plumejeaud**

en vue d'obtenir

LE DIPLÔME D'INGENIEUR C.N.A.M.

en INFORMATIQUE

Acquisition de données et cartes de potentiel pour l'analyse spatiale

Soutenu le 28 Juin 2007

Les travaux relatifs à ce mémoire ont été effectués au laboratoire LIG (anciennement LSR-IMAG) sous la direction de M. Jérôme Gensel.

SOMMAIRE

FIGURES

TABLEAUX

ABREVIATIONS

API *Application Programming Interface*, ou Interface pour la Programmation d'Applications. Ensemble de bibliothèques permettant une programmation plus aisée car les fonctions deviennent indépendantes du matériel. On peut citer par exemple les API de DirectX ou de Java.

ASCII *American Standard Code for Information Interchange*, norme de codage de caractères en informatique la plus connue et largement compatible. Définissant 128 caractères, codés sur 7 bits, elle suffit pour écrire en anglais. Un fichier ASCII est donc composé de suite de caractères lisibles.

CVS *Concurrent Versions System*, système de gestion de versions de fichiers numériques à accès concurrent. Il est couramment utilisé pour conserver et manipuler les sources d'un logiciel dans un environnement collaboratif.

ESPON *European Spatial Planning Observation Network*, organisme de la commission européenne qui s'est donné pour mission l'observation et le conseil pour l'aménagement du territoire européen. Site Web: http://www.espon.eu/

GIF *Graphics Interchange Format*, littéralement « format d'échange de graphiques » a été mis au point par CompuServe en 1986 pour permettre l'échange d'images numériques sur le Web, en utilisant l'algorithme de compression LZW, nettement plus efficace que RLE (utilisé par PCX, BMP, etc.).

HTML HyperText Markup Language, langage de balisage de texte qui permet la création de documents hypertextes affichables par un navigateur Web. [@GrandDico]

HTTP *HyperText Transfer Protocol*, littéralement « protocole de transfert hypertexte », est un protocole de communication informatique client serveur développé pour le World Wide Web. Il est utilisé pour transférer les documents (document HTML, image, feuille de style, etc.) entre le serveur HTTP et le navigateur Web lorsqu'un visiteur consulte un site Web.

HTTPS *HyperText Transfer Protocol Secure*, qualifie le protocole HTTP sécurisé par cryptage des communications via SSL.

JTS *Java Topology Suite*, bibliothèque de composants Java permettant de manipuler des formes géométriques respectant la norme OpenGIS.

JVM *Java Virtual Machine*, la machine virtuelle Java est un moteur d'exécution permettant d'interpréter le langage binaire (*bytecode*) produit par la compilation de code Java.

JPEG *Joint Photographic Experts Group*, norme de compression d'images. Cette norme permet de compresser les images, mais selon le taux de compression, l'image perd en qualité. Le JPEG offre un nombre de couleurs affichables plus vaste que le GIF et permet notamment les dégradés de couleurs.

MIF/MID *MapInfo Interchange Format*, format standard de fichiers contenant des informations géographiques, conçu par la société MapInfo. Le couple de fichiers au format ASCII contient pour l'un, le fichier .MIF des fonds de cartes (système de projection et suite d'objets géométriques), et pour l'autre, le fichier .MID, un ensemble d'informations portées par ces objets, dont au moins un identifiant unique.

MVC *Model View Controler*, cadre de développement pour des applications Web proposant de séparer les données métier des objets servant à la visualisation, et d'instaurer un contrôle des interactions entre les deux couches précédentes dans une couche logicielle dédiée.

NUTS Nomenclature des Unités Territoriales Statistiques, harmonisation des maillages administratifs des différents pays européens sur une échelle à 5 niveaux : pays (0), grandes régions (1), régions (2), départements (3), districts (4), communes (5). [@Europa]

OGC *Open Geospatial Consortium*, organisation internationale à but non lucratif visant à établir des normes « OpenGIS » pour la production de services géo-localisés.

PCX PCX est un format bitmap permettant d'encoder des images dont la dimension peut aller jusqu'à 65536 par 65536 et codées sur 1 bit, 4 bits, 8 bits ou 24 bits. Il était très populaire sur les premiers systèmes DOS et Windows, mais il se fait de plus en plus rare, car il existe de nos jours des formats permettant une compression bien meilleure, comme par exemple GIF, JPEG et PNG.

PNG *Portable Network Graphics*, format d'images numériques ouvert, qui a été créé pour remplacer le format propriétaire GIF, dont la compression était soumise à un brevet. Le PNG est un format non destructeur spécialement adapté pour publier des images simples comprenant des aplats de couleurs.

RMI *Remote Method Invocation*, mécanisme Java pour l'invocation transparente de méthodes sur des objets distants (c'est-à-dire des objets instanciés sur une autre machine virtuelle) mais accessibles via un réseau de communication informatique (Internet par exemple).

RPC *Remote Procedure Control*, protocole permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'application. Ce protocole est utilisé dans le modèle client-serveur et permet de gérer les différents messages entre ces entités.

SCM *Source Content Management*, système dédié à la gestion de versions de sources d'un logiciel, dans des environnements de développement collaboratif. CVS, Subversion, ClearCase en sont des exemples.

SIG Système d'Information Géographique, permet de gérer, analyser et diffuser des données localisées (points géographiques) et les informations qui leur sont associées.

SGBD Système de Gestion de Base de Données

SLK Symbolic Link Interchange, format d'export du logiciel tableur MULTIPLAN, le prédécesseur d'Excel, commercialisé par Microsoft dans les années 80. Format texte compris par Excel.

SOAP *Simple Object Access Protocol*, méthode inventée par Microsoft pour faire du RPC sur Internet via des requêtes HTTP. SOAP est aujourd'hui une recommandation du W3C implémentée par le projet Apache, spécifiant un protocole qui définit une manière formelle de faire transiter des données au format XML. [@W3C-SOAP]

SQL *Structured Query Language*, language informatique d'interaction avec un système de base de données relationnelles permettant d'interroger, insérer ou supprimer des données.

SSH *Secure Shell*, protocole permettant à un client d'ouvrir une session interactive sur une machine distante (serveur) afin d'envoyer des commandes ou des fichiers de manière sécurisée, par chiffrage des données et authentification mutuelle.

SSL *Secure Socket Layer* ou couche de sockets sécurisée est un procédé de sécurisation des transactions effectuées via Internet. Il repose sur un procédé de cryptographie par clef publique afin de garantir la sécurité de la transmission de données. Il est indépendant du protocole utilisé.

SVG *Scalable Vector Graphics*, spécification du W3C permettant de définir des graphiques vectoriels à partir d'une grammaire XML spécialisée [SVG, 2003]

UML *Unified Modelling Language*, méthode de conception et de représentation de systèmes à objets [Booch et al., 2000].

W3C *World Wide Web Consortium*, abrégé W3C, est un consortium fondé en octobre 1994 pour promouvoir la compatibilité des technologies du Web telles que HTML, XHTML, XML, CSS, PNG, SVG et SOAP. Le W3C n'émet pas des normes, mais des recommandations.

XML *eXtensible Markup Language*, évolution du langage SGML permettant aux concepteurs de documents HTML de définir leurs propres marqueurs, dans le but de personnaliser la structure des données qu'ils comptent présenter. C'est en utilisant un navigateur compatible que l'utilisateur peut exploiter les marqueurs personnalisés du langage XML. De ce fait, ce langage est mieux adapté à la gestion de documents longs et complexes, comme on en trouve dans les intranets, puisque l'utilisateur peut sélectionner le type d'information qu'il souhaite consulter.

Remerciements

En tout premier lieu, je remercie toutes les personnes qui me font l'honneur de participer au jury de ce mémoire : merci donc à Madame Véronique Donzeau-Gouge et Monsieur Eric Gressier-Soudan, professeurs au CNAM Paris ; à Monsieur Jean-Pierre Giraudin, professeur à l'Université Pierre Mendès-France de Grenoble ; à Monsieur Jean-Marc Vincent, Maître de Conférences à l'Université Joseph Fourier, à Monsieur Jean-Claude Fernandez professeur à l'Université Joseph Fourier ainsi qu'à Monsieur André Plisson, Directeur du centre d'enseignement CNAM de Grenoble.

Je tiens à exprimer ma reconnaissance à mon tuteur, Jérôme Gensel, Maître de Conférences à l'Université Pierre Mendès-France de Grenoble, pour ses fréquentes relectures et ses conseils. Je remercie aussi le géographe Claude Grasland, et la cartographe Hélène Mathian pour le suivi attentif de mes travaux et la passion qu'ils savent communiquer. J'exprime aussi ma gratitude à Jean-Marc Vincent dont le niveau d'exigence scientifique nous amène à tirer le meilleur de nous-même. Merci aussi à Marlène Villanova, Hervé Martin et Paule-Annick Davoine, membres permanents de l'équipe STEAMER pour leur sympathie constante.

J'ai une attention spéciale pour Guillaume Vergnaud, thésard géographe au laboratoire Géophile de Lyon, grâce à qui HyperCarte bénéficie maintenant d'un jeu de données sur la région Rhône-Alpes et qui a motivé mes travaux sur le module d'acquisition de données. De même, pour Serge Guelton, ingénieur de l'équipe MESCAL, qui a largement contribué au développement d'un module de lissage cartographique viable. Je n'oublie pas Christophe Chabert, mon prédécesseur, qui a consacré trois semaines de son temps de mémoire à m'expliquer de façon très pédagogique le projet.

Je remercie les thésards du Laboratoire d'Informatique Grenoblois (LIG), du troisième étage du bâtiment D de l'ENSIMAG, pour leur compagnie prévenante et chaleureuse, et leur bonne humeur. En particulier Alina Dia Miron, qui m'a fait découvrir la Roumanie par monts et par vaux. Merci à Yan Grunenberger pour ses critiques constructives et son regard scientifique sur mon travail.

L'amitié et le sérieux des membres de l'association AIPST CNAM furent aussi une boussole. Leurs critiques constructives, et leur expérience m'ont soutenue pendant mon mémoire. Je remercie en particulier les membres du bureau.

Ma gratitude va à mes amis qui ont su m'entourer pendant cette période difficile et mouvementée.

Enfin, je dédie ce mémoire à mon père qui aurait certainement été fier de l'accomplissement de mon projet professionnel. Merci papa.

1. Introduction

1.1. Contexte

La cartographie nous donne aujourd'hui les moyens de mieux appréhender le monde qui nous entoure. Ainsi des logiciels, tels que GoogleMap, proposent une navigation en trois dimensions sur un globe virtuel, permettant de survoler le relief, les villes, de visualiser un lieu touristique, de localiser notre maison, etc. Cependant, notre environnement ne se limite pas à des objets géographiques réels : nous appartenons à une nation, nous faisons partie d'une catégorie professionnelle, d'une tranche d'âge, nous avons un statut familial, etc., et ces éléments nous situent dans la société. Il s'agit de donner une représentation spatiale à ces statistiques établies à partir du recensement d'individus sur des entités géographiques, le plus souvent des unités administratives : villes, départements, régions, pays.

Par ailleurs, ces statistiques font l'objet d'une analyse exploratoire permettant de déterminer des tendances dans leur répartition spatiale, que l'on nomme communément analyse spatiale. Cette science, qui permet par exemple de donner une illustration de la distribution des richesses dans le monde, a bénéficié ces dernières années des énormes progrès de la cartographie, aussi bien sur le plan conceptuel que sur le plan technique. En effet, la multiplication des techniques de représentation, et la rapidité des traitements que l'on peut effectuer sur les données ont permis de vérifier et de concevoir de nouvelles hypothèses concernant la structure spatiale des phénomènes étudiés. En retour, les statisticiens et géographes ont élaboré des méthodes de calcul et des modèles mathématiques plus complexes, afin d'affiner les analyses.

Egalement, l'analyse spatiale s'ouvre au grand public car les avancées dans le domaine du Web ont ouvert de nouvelles perspectives pour la diffusion et l'établissement des cartes. En effet, l'utilisateur n'est plus limité à la lecture de cartes figées, il peut paramétrer son analyse et générer lui-même la représentation qui lui convient. Cette interactivité ne nécessite plus d'être propriétaire des données, ou bien spécialiste du domaine étudié.

1.1.1. Le groupe de recherche Hypercarte

Ce mémoire est une contribution aux travaux du groupe de recherche *HyperCarte*, qui entend fournir des outils permettant une visualisation et une analyse spatiale interactive de phénomènes socio-économiques. *HyperCarte* est un groupement de recherche fondé en 1996 et ayant à son actif de nombreux travaux et réalisations scientifiques et une reconnaissance nationale et internationale dans le domaine de la cartographie interactive.

Le projet *HyperCarte* est basé une coopération étroite entre diverses équipes de recherche issues de disciplines des Sciences Humaines et Sociales (géographie, sociologie, économie) et des équipes de recherche en informatique.
Les objectifs du projet *HyperCarte* sont :

- La conception et la réalisation de modules d'analyse cartographique fondamentaux, cœur d'une plate-forme logicielle, facilement accessible par des utilisateurs aux profils divers (experts géographes, statisticiens, politiques, grand public, etc.).

- L'implémentation d'applications thématiques de ces modules, afin de répondre à des demandes sociales ou politiques précises, notamment dans le domaine de l'aménagement du territoire européen (projet *Espon-Hyperatlas* de l'Union Européenne) ou dans celui de l'étude des inégalités économiques et sociales au niveau mondial (*Hypercarte-Monde*).

Exploitant un ensemble de données classiques (indicateurs démographiques ou socio-économiques de base), un serveur cartographique *HyperAtlas* permet la production à la volée d'un ensemble de cartes interactives paramétrables par l'utilisateur. En effet, la conviction du groupe de recherche *HyperCarte* est qu'il n'existe pas une représentation cartographique unique d'un phénomène social, mais un très grand nombre, selon la nature intrinsèque des phénomènes sociaux, environnementaux, etc., en fonction des hypothèses du concepteur de la carte, des objectifs, des demandes, des pratiques ou des croyances des utilisateurs finaux de l'information cartographique [Andrienko, 1999]. L'évolution des technologies associées au Web offre d'énormes possibilités à ce type d'approche, en particulier grâce à la souplesse qu'apporte une interactivité de haut niveau [Kraak, 2000]. L'enjeu est aussi de cibler simultanément un objectif de cartographie de communication "grand public" et une cartographie plus "exploratoire" stimulant la réflexion.

Le nombre de cartes possibles pour décrire un même phénomène étant virtuellement infini, le serveur cartographique réalisé jusque là est un outil de conception à la demande reposant sur une triple compétence :

1. *compétence en géographie et sciences sociales* : pour définir les phénomènes potentiellement intéressants et les types de traitements à leur appliquer. Les problèmes théoriques posés par l'analyse spatiale et la cartographie des phénomènes sociaux constituent le premier axe de travail et relèvent plus directement de la responsabilité de l'équipe *PARIS* de *l'Unité Mixte de Recherche (UMR) Géographie-cités*, en partenariat avec d'autres équipes ou réseaux de recherche en sciences sociales. Il s'agit de faire l'inventaire des avancées récentes de la réflexion sur l'étude spatiale des formes sociales. L'idée centrale est ici d'exploiter les méthodes statistiques et cartographiques habituellement utilisées pour étudier des phénomènes spatiaux quelconques et étudier dans quelle mesure ils peuvent s'appliquer au problème plus spécifique des phénomènes sociaux.

2. *compétence en informatique distribuée* : pour gérer de façon optimale les flux de demandes, le traitement numérique de grands volumes de données, ainsi que les caches susceptibles d'assurer un temps de réponse rapide. Les problèmes informatiques posés par la mise au point de modules cartographiques automatiques sur serveur Web relèvent davantage de la responsabilité de l'équipe d'informatique et de calcul parallèle, l'équipe *MESCAL* du *Laboratoire d'Informatique de Grenoble* (*LIG)*, (ex *ID-IMAG*), qui propose une infrastructure de type grappes de PC permettant le calcul en temps réel des cartes, mais également des pré-calculs et stockage intermédiaire aptes à répondre à un flux de demandes élevé, en évitant le stockage de l'ensemble des cartes possibles.

3. *compétence en systèmes d'information et multimédia* : pour produire des cartographies interactives des phénomènes socio-économiques, adaptées à différents profils

d'utilisateurs. Les problèmes informatiques posés par l'accès à une information virtuellement infinie et par l'hétérogénéité des utilisateurs potentiels relèvent quant à eux de l'expertise de l'équipe *STEAMER* du laboratoire *LIG*, (ex LSR-IMAG), qui doit mettre au point un système d'information apte à délivrer rapidement aux utilisateurs les cartes les plus adaptées à leur besoin, tout en veillant à ce que la restitution de cette information ne soit pas source d'ambiguïtés ou d'erreurs d'interprétation de la part de l'utilisateur. Il faut donc détecter les profils des utilisateurs, soit par une déclaration directe de ces derniers (fiche d'identification), soit par une observation indirecte de leurs requêtes (repérage de profils-type) et ensuite fournir des options, des aides en lignes, des commentaires de résultats les mieux adaptés aux utilisateurs.

Le travail que nous rapportons ici se situe au confluent de ces trois exigences : basé dans le laboratoire du *LIG*, au sein de l'équipe *STEAMER*, nous sommes en charge de coordonner les trois visions du projet pour mener à bien sa réalisation. Ce mémoire fait suite à un probatoire dont le sujet portait sur le recensement et la catégorisation des outils disponibles pour la cartographie libre sur le Web, en novembre 2005 [Plumejeaud, 2005]. Cette base de connaissances thématiques nous permet d'aborder la problématique du sujet en nous concentrant sur les difficultés techniques qu'il met en jeu.

1.1.2. Une réalisation d'Hypercarte : HyperAtlas

A l'heure actuelle, le projet Hypercarte a réalisé un outil interactif de représentation cartographique de phénomènes divers (socio-économiques, environnementaux, culturels…) référencés dans un espace ou territoire : *HyperAtlas*. L'application propose une analyse territoriale multiscalaire par la visualisation des valeurs d'indicateurs pour les unités territoriales qui composent le maillage de l'espace étudié. Ce maillage (administratif, par zones d'activité, zones d'attraction, etc.) peut comporter différents niveaux d'imbrication : une unité territoriale – une maille – peut elle-même être composée de plusieurs unités territoriales. Un maillage définit une hiérarchie d'unités territoriales représentée par un arbre (cf. figure 1) dont :

- Le noeud racine correspond à l'unité territoriale englobante recouvrant l'espace d'étude (par exemple, l'Europe),
- les nœuds feuilles sont les unités territoriales les plus petites (par exemple, pour le maillage NUTS, les communes en France),
- les autres nœuds représentent les unités territoriales intermédiaires (par exemple, pour le maillage NUTS, les régions d'Europe).

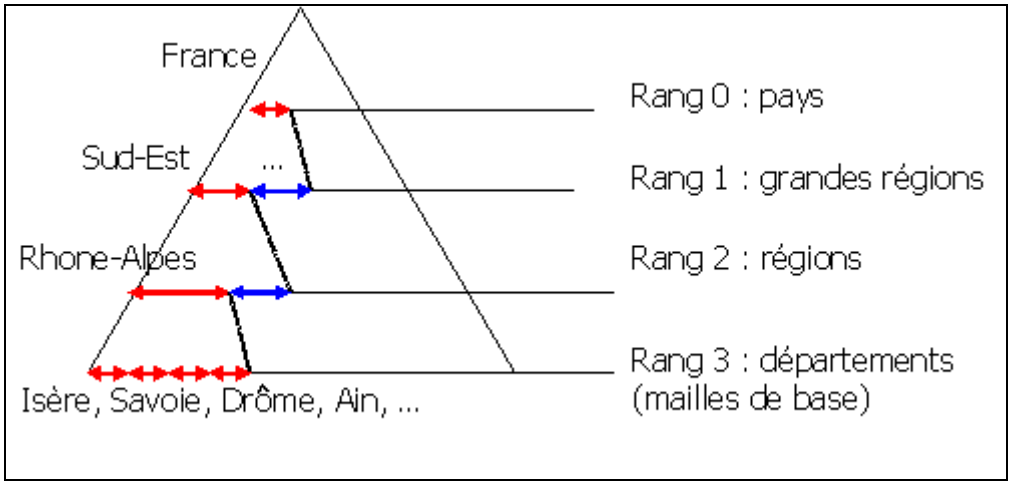


Figure 1. Arbre figurant la hiérarchie des maillages du NUTS.

L'application met en œuvre des comparaisons de ratios d'indicateurs sur différentes échelles. En parcourant l'arbre verticalement, le ratio d'une unité est rapporté à celui d'une unité englobante, par le calcul d'une déviation globale, ou moyenne. Dans le sens horizontal, ce ratio est comparé à celui d'une unité de même niveau hiérarchique entretenant une relation de proximité, par le calcul d'une déviation locale. Les cartes résultantes de cette analyse (d'indicateurs, de ratios, de déviations) offrent à l'utilisateur l'opportunité d'évaluer une donnée thématique dans divers contextes, qualifiant ainsi l'analyse de multiscalaire.

L'application *HyperAtlas*, réalisée initialement dans le cadre du programme ESPON avec l'Union Européenne, permet aujourd'hui de visualiser différents jeux de données, que ce soit sur l'Europe, le Cameroun, ou la Tunisie. En effet, les fonctionnalités ne sont pas liées à un espace particulier et peuvent donc être implémentées sur des jeux de données différents, dès lors qu'il s'agit de données fondées sur un maillage politique ou administratif comportant des relations de contiguïté et d'emboîtement.

Par exemple, avec le jeu de données Européen, l'interface permet la visualisation de cartes rendant compte de la distribution de différents indicateurs socio-économiques comme la population, le Produit National Brut (PNB ou GDP), ou le taux de chômage (cf. figure 2), et ceci sur diverses aires d'étude : l'Europe des 15, des 25, des 29 (incluant les deux nouveaux pays membres que sont la Roumanie et la Bulgarie), l'Arc Atlantique, etc. Les maillages disponibles appartiennent au NUTS, la nomenclature européenne officielle : du niveau 0 (pays) au niveau 3 (équivalent des départements français).

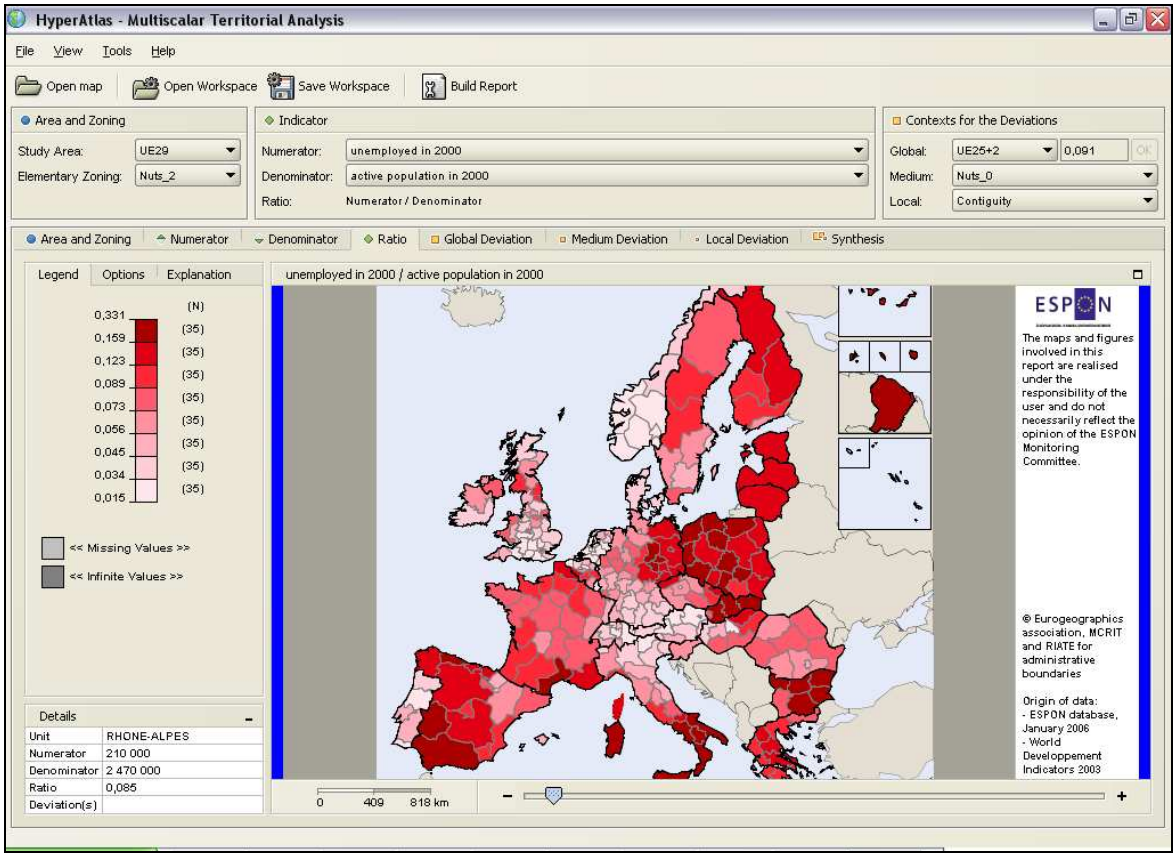


Figure 2. Interface de *HyperAtlas* ESPON - carte du ratio visualisant le taux de chômage en Europe (EU29) en 2000, sur les régions (NUTS 2).

Le logiciel existe en deux versions : une première version standard (sans personnalisation) permet de visualiser n'importe quel jeu de données, et une seconde version labellisée aux couleurs de ESPON (*ESPON HyperAtlas*) est fournie avec seulement un jeu de données sur l'Europe dont ESPON est co-propriétaire. Il est important de noter que la licence du logiciel, mise à jour en Avril 2006, précise que le programme ESPON ne possède qu'un droit de copropriété sur l'application ESPON *HyperAtlas* et que les équipes de recherche engagées dans le projet *HyperCarte* gardent toute liberté pour réaliser d'autres implémentations du logiciel sur d'autres jeux de données.

1.2. Problématique : les enjeux du projet

HyperAtlas, dont la réalisation a débuté en 2003 avec la reprise d'un premier prototype par Philippe Martin [Martin, 2004], et sur lequel ont travaillé successivement Olivier Cuenot [Cuenot, 2005] et Christophe Chabert [Chabert, 2007], a atteint un stade de maturité suffisant : les concepteurs géographes jugent que le résultat est satisfaisant, et en dehors d'une maintenance évolutive, aucune évolution fondamentale n'est prévue cette année. Par ailleurs, le programme ESPON a permis de tester les performances du logiciel et d'y apporter de nombreuses améliorations en termes d'interface utilisateur ou de réponse aux besoins en matière d'aide à la décision politique en matière d'aménagement du territoire. Après avoir financé en 2003-2004 un premier prototype dans le cadre du projet ESPON 3.1

(*Integrated Tools*), le programme ESPON a demandé une évaluation auprès de 120 utilisateurs scientifiques et politiques répartis dans 29 pays européens. Les résultats de l'évaluation ont été très positifs et le programme ESPON a apporté un complément de financement en 2005-2006 pour aboutir à une version finale opérationnelle qui est téléchargeable sur son site et fait désormais partie de l'acquis communautaire en matière de recherche sur les inégalités régionales. Fort de cette maturité, le projet a décidé de protéger le logiciel en s'adressant à l'Agence pour la Propriété des Programmes (APP). Les spécifications ainsi que l'ensemble du code source (datant de Mars 2006) sont déposées à l'APP. L'application a fait l'objet d'une communication au colloque de Géomatique SAGEO 2005 [Grasland, 2005a] et peut être téléchargée sur le site[1] du programme ESPON (dans une version labellisée aux couleurs d'ESPON).

Mais dans le cadre de sa maintenance, il apparaît une grande lacune dans le domaine de la gestion des sources et le suivi des évolutions du logiciel : il n'existe pour ainsi dire aucune gestion de configuration, ni recensement formel des erreurs (*bugs*), ni suivi de leur traitement. Et la documentation est très mal gérée, dispersée entre les trois équipes, il est difficile même de connaître l'existence de tel ou tel document, tout reposant sur la connaissance des permanents du projet. De cette observation jaillit un besoin fort en matière de gestion et d'organisation du projet.

D'autre part, l'application présente certaines lacunes par rapport aux logiciels équivalents. Par exemple, beaucoup d'usagers d'HyperAtlas regrettent de ne pas avoir la possibilité d'acquérir et gérer leurs propres données, comme le proposent d'autres logiciels. Leur accorder cette possibilité aurait certainement un effet bénéfique sur la diffusion du logiciel, et augmenterait le champ d'expérimentations possibles sur les jeux de données. Un premier prototype existe mais il nous faut l'améliorer sur plusieurs plans.

Par ailleurs, HyperAtlas n'offre par exemple aucune forme de représentation continue des phénomènes analysés. Or ce type de représentation permet de lire l'organisation spatiale de phénomènes sociaux-économiques, en affranchissant le lecteur des maillages spatiaux, qui ne correspondent que peu souvent aux discontinuités réelles du phénomène. Le projet Hypercarte a élaboré depuis plusieurs années une méthode pour ce genre de représentation - la méthode du potentiel - et désire tester sa mise en œuvre dans le cadre d'une architecture de calculs distribués. En effet, cette méthode est assez coûteuse en temps calcul, et réaliser une interface aussi interactive qu'HyperAtlas pour paramétrer l'analyse est un défi technologique.

1.3. Contribution

Nous partons d'une précédente réalisation, *HyperAtlas*, dont nous devons assurer la maintenance et faire évoluer conformément aux besoins exprimés par les experts et les usagers. En effet, le projet a obtenu une subvention du programme européen pour l'aménagement du territoire, ESPON, et la satisfaction du contrat nécessite d'apporter quelques améliorations, et de régler certaines anomalies de fonctionnement.

Nous héritons de plus d'un module baptisé *HyperAdmin* destiné à l'acquisition de données, dans une version prototype (V0.1 Alpha 2), et qui, bien que remplissant

[1] *http://www.espon.eu/mmp/online/website/content/tools/912/index_EN.html*

partiellement le cahier des charges exprimé, ne répond pas à l'attente des utilisateurs sur plusieurs plans. En conséquence, nous devons diagnostiquer l'application pour déterminer comment faire évoluer l'existant. L'objectif étant de satisfaire les utilisateurs, mais aussi de constituer un socle de développement durable et suffisamment robuste pour supporter les évolutions à venir.

Enfin, depuis plusieurs années, le projet consolide une théorie sur la représentation continue des phénomènes spatiaux basée sur l'utilisation de cartes de potentiel. Il souhaite mettre ses résultats en application dans un module au moins aussi ergonomique, fiable et interactif que *HyperAtlas*. Cette réalisation doit se faire en collaboration étroite avec l'équipe MESCAL du LIG car elle met en œuvre des algorithmes nécessitant de distribuer les calculs sur des machines multiprocesseurs. Or, dans les deux mois suivants le début de ce travail, l'équipe MESCAL aura à sa disposition durant 5 mois un ingénieur qui se consacrera à ce module, du 15 mai au 15 septembre 2006. Cette période nécessite donc un investissement important, quasiment exclusif sur ce module que l'on appelle *HyperSmooth.*

Une part non négligeable de notre action consiste à donner une structure solide au projet sur le plan de l'ingénierie logicielle, pour améliorer notre efficacité et faciliter la maintenance des 3 modules à venir. Pour cela nous harmonisons et rationalisons les méthodes de développement.

1.4. Plan du mémoire

Après ce chapitre introductif, nous structurons en cinq chapitres ce mémoire, présentant respectivement un état de l'art, une proposition en trois chapitres, et une conclusion.

Notre état de l'art vise à dresser un panorama des technologies disponibles pour faire de la cartographie interactive sur le Web, et met en perspective diverses formes de représentation des phénomènes sociaux-économiques. A partir du croisement de ces deux aspects, technologique et sémantique, nous répertorions et critiquons les outils d'analyse spatiale les plus similaires à HyperAtlas. En particulier, nous précisons leur offre en matière de gestion et d'acquisition de données.

Notre proposition est découpée en trois chapitres, dont le premier constitue un préambule aux deux suivants : il décrit de façon générale les besoins auxquels doivent répondre les modules *HyperAdmin* et *HyperSmooth*, et présente succinctement le fonctionnement de HyperAtlas et son architecture. Cette présentation permettra au lecteur de comprendre les choix technologiques ainsi que les détails de l'implémentation des deux prochains modules. Ensuite, le chapitre 4 est entièrement dédié à *HyperAdmin*, et le chapitre 5 à *HyperSmooth*. Ces deux chapitres ont une structure similaire : cahier des charges, proposition technique, réalisation et bilan.

Enfin, nous concluons, en synthétisant les bilans dressés sur notre réalisation dans chacun des modules, ainsi que sur notre apport global en matière de gestion technique du projet. Nous présentons alors aussi un bilan personnel de ce mémoire.

2. Etat de l'art

Cet état de l'art présente les outils aujourd'hui disponibles pour une cartographie dynamique sur le Web. Nous mettons aussi en perspective diverses formes de représentation de l'information en analyse spatiale. Puis, nous croisons ces deux aspects pour apporter notre point de vue sur les outils cartographiques existant à l'heure actuelle et dédiés à l'analyse spatiale.

2.1. Les outils d'une cartographie dynamique sur le Web

Nous désirons ici présenter le vocabulaire communément utilisé en géomatique, et aider à mieux comprendre le fonctionnement d'un logiciel permettant l'édition de cartes.

2.1.1. Principes généraux

En premier lieu, tous ces logiciels reposent sur l'utilisation de Systèmes d'Information Géographique (SIG). Un SIG est un système informatisé de stockage, d'analyse et de recherche de l'information dans lequel toutes les données sont classées sur une base spatiale en fonction de leurs coordonnées géographiques. Les données géoréférencées sont des variables thématiques qui peuvent être de natures variées : l'altitude, le nom du lieu, la température, le nombre d'habitants, etc.

La donnée peut être gérée et stockée de plusieurs manières :

- La donnée peut être gérée grâce à un Système de Gestion de Base de Données (SGBD).
- La donnée peut être conservée sous forme de fichier SIG. Les shapefiles d'extension « .shp » sont un exemple de fichier généré par ArcInfo[2]. Les fichiers générés par MapInfo[3] portent l'extension « .mif » et nous les nommons par le sigle MIF. Ces deux formats de stockage et d'échange de données sont très répandus.
- La donnée peut être incluse dans un script.

Un SIG n'est pas simplement une base de données ; c'est aussi un outil d'analyse et de visualisation des données. Cette visualisation nécessite une opération de projection des coordonnées sphériques dans un système de représentation plan. Cette projection est calculée par le SIG, qui reconstitue ensuite une image à partir des données.

a) Modes de visualisation

L'image peut être rendue dans deux modes différents. Le mode image ou matriciel (***raster***) est le plus ancien, c'est aussi le plus robuste. Nativement, un navigateur Web connaissant le langage HTML peut afficher une image numérique, encore appelée image Bitmap. Une image numérique se compose d'une matrice de points élémentaires, les *pixels*, auxquels sont associés une couleur et une luminosité. Pour faire le parallèle avec les mathématiques, c'est une définition de l'image en extension. L'inconvénient de ce mode est qu'il ne permet pas une grande interactivité. De plus la taille en octets d'une image Bitmap

[2] SIG commercialisé par la société ESRI, http://www.esri.com/
[3] SIG commercialisé par la société MapInfo, http://www.mapinfo.com/

augmente considérablement avec ses dimensions et sa définition. Ce qui n'est pas le cas pour l'image vectorielle.

Le second mode, le mode **vecteur**, est plus riche en informations. Les fichiers vectoriels contiennent une description des entités géométriques à représenter : points, lignes, surfaces, formes géométriques élémentaires, indication de couleurs, etc. C'est en quelque sorte une définition de l'image en compréhension, comme lorsqu'on définit en mathématiques l'ensemble des nombres entiers pairs par la formule suivante : $\{x \mid x = 2*k, k \in IN\}$. Il a l'avantage de générer des images de meilleure qualité que les images Bitmap. Cependant, ce mode nécessite d'utiliser un logiciel complémentaire côté client, qu'on appelle *plugin*. Il existe différents formats vectoriels, nous le verrons par la suite.

b) Traitement des couches de données

Un SIG peut être utilisé pour traiter divers types et formats de données. Celles-ci peuvent prendre la forme de plusieurs **couches** différentes où chaque couche contient des données pour un type d'entité particulier (bâtiments, rues, relief, etc.). A chaque couche de données, nous pouvons associer un calque image différent.

Le SIG, ou alors l'outil de visualisation, est capable de **fusionner** ces calques pour former l'image finale. Pour cette fusion, il doit tenir compte des différences d'échelle qu'il peut exister entre chaque couche de données.

En effet, selon le type de données, l'échelle peut être différente. On peut, de plus, devoir fusionner des cartes utilisant des repères différents, étant donné qu'il n'existe malheureusement pas de standard mondial assurant la cohérence des cartes.

c) Diffusion sur Internet

Internet est le nom donné au réseau informatique mondial, reposant sur le système d'adresses global des protocoles de communication *Transmission Control Protocol/Internet Protocol (*TCP/IP) et qui rend accessible au public des services comme le courrier électronique et le World Wide Web (Web). Les services Web reposent sur une architecture client/serveur où :

- les **clients** sont des programmes logiciels demandeurs d'information. Les plus communément utilisés sont les navigateurs Web, Internet Explorer, Mozilla, Opéra, Netscape : ils savent au minimum interpréter les réponses rendues au format *Hyper Text Markup Langage* (HTML). Si on les équipe de *plugin*, ils peuvent alors interpréter des formats plus spécifiques.
- les **serveurs** sont des ordinateurs équipés de logiciels particuliers capables de répondre à des questions émises par les clients, si la question fait partie du type de service proposé. Ils sont toujours en attente des demandes des clients. On dit que le serveur retourne une page Web au client, en réponse à une requête HTTP. HTTP est le protocole de communication entre les serveurs et les clients.

Ces pages peuvent être de deux types : statiques ou dynamiques. Une page statique ne peut pas être personnalisée selon l'utilisateur, car elle a été générée et stockée sur le serveur avant le lancement du serveur. Ce type de site, dit statique, n'est pas connecté à une base de données. Les **pages dynamiques** sont construites à la volée, et peuvent être adaptées en fonction des demandes utilisateur. Ces sites sont alors connectés sur des bases de données. La

plupart des sites cartographiques sont dynamiques : ils génèrent la carte en fonction des paramètres passés dans la requête http du client.
La figure 3 synthétise l'architecture Web classique telle que nous venons de la décrire.

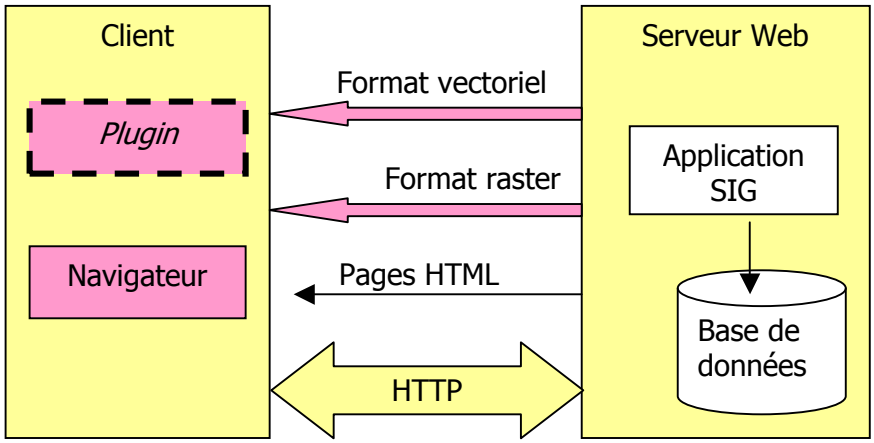


Figure 3 : Architecture classique d'un service Web de cartographie

A noter, et nous le verrons par la suite, que ce n'est pas l'unique architecture possible. On peut par exemple héberger la base de données sur une machine différente de celle qui héberge le serveur Web. Le client peut aussi être équipé d'un module applicatif (*plugin*, *applet ou* visionneuse) effectuant une bonne partie du traitement des données pour fabriquer l'image.

2.1.2. Revue des technologies

Nous abordons les techniques les plus couramment utilisées en matière de cartographie interactive multimédia pour la diffusion de cartes sur le Web, supportant la diffusion de formats vectoriels et matriciels.

a) Pour une cartographie animée

Nous exposons ici des formats qui offrent un haut niveau d'interactivité au client, mais qui requièrent l'implantation d'interpréteurs spécifique (des *plugins*) sur le navigateur. Puis une nouvelle technologie, Ajax, qui bien que ne nécessitant pas l'installation de *plugin*, n'est supportée que dans les navigateurs Web, les plus récents.

- **SVG**

SVG a été conçu par le W3C[4] afin de proposer un standard en termes de graphisme vectoriel. Il s'agit d'un dialecte XML qui peut être édité avec un éditeur texte. Un autre avantage hérité de XML est de s'intégrer parfaitement dans la hiérarchie des objets d'une page Web, que l'on appelle le Modèle d'Objet de Document (DOM) [Neumann, 2003]. On peut donc appliquer des fonctions Javascript aux objets définis en SVG.

[4] Le World Wide Web Consortium, abrégé W3C, est un consortium fondé en octobre 1994 pour promouvoir la compatibilité des technologies du Web.

SVG permet de définir des primitives (rectangle comme dans la figure 4, ellipse, courbe de Bézier, forme librement définie, etc.), des effets de style (transparence, dégradé, etc.), des transformations géométriques (changement de dimension, rotation, translation), des interactions avec l'utilisateur (clic ou survol de souris, touche de clavier enfoncée) et des animations.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE svg SYSTEM "svg10.dtd">
<svg>
   <rect x="5" y="5" height="100" width="150" style="fill:yellow"/>
</svg>
```


Figure 4. Exemple : Dessiner un rectangle jaune uni en langage SVG.

L'utilisateur, par interaction, peut déclencher des animations, mais ces dernières ont la possibilité d'être scénarisées et déclenchées en fonction du temps. A la manière de la balise `<img/>` du XHTML, SVG permet d'inclure dans un document des références à des fichiers images en mode point ou à des fichiers SVG. Des transformations géométriques et des modifications de style peuvent être appliquées à ces éléments inclus.

La taille des fichiers SVG peut être réduite de 50 à 80% grâce à l'algorithme de compression gzip (ils possèdent alors l'extension .svgz). SVG est de mieux en mieux supporté par les SIG, qui permettent d'exporter au format SVG, ou de visualiser des documents SVG.

- **Flash**

Flash a été conçu par la société Macromédia en 1996. C'est aujourd'hui un standard de fait, alors qu'il n'est avalisé par aucun organisme de standardisation. Il s'agit d'un format binaire propriétaire. Il est utilisé largement par l'industrie multimédia et les publicitaires pour créer des dessins et des animations en mode vectoriel.

Flash autorise un téléchargement progressif. Par exemple, une carte peut être affichée après un téléchargement minimum, et les données permettant d'afficher les détails n'être téléchargées que lorsque l'utilisateur zoome sur la carte. Le format de fichier utilisé par Flash est fortement compressé, ce qui réduit les temps de téléchargement des fichiers, mais peut rendre l'indexation du contenu de ces derniers difficile.

L'animation et l'interactivité avec l'utilisateur sont obtenues grâce à un langage de script qui permet au programmeur d'orchestrer des scènes – c'est-à-dire de construire des scènes chrono datées, et de gérer le survol d'éléments, la sélection de couches, etc. Flash permet d'interroger des bases de données. En cartographie, cela permet, par exemple, d'interroger des bases de données réparties contenant chacune les données d'une couche de cartes.

Le format d'impression est vectoriel, c'est-à-dire que les informations envoyées à l'imprimante sont, non pas en mode point, mais de type vectoriel.

- **Les applets Java**

Java est une technologie composée d'un langage de programmation objet et d'un environnement d'exécution. L'originalité de Java vient de sa portabilité : les programmes écrits en Java peuvent s'exécuter sur plusieurs types de matériel informatique. L'environnement d'exécution, la Java Virtuelle Machine (JVM), doit être installé préalablement sur la machine qui exécutera le code. La JVM est ensuite capable d'interpréter le code Java compilé (le *byte-code*). Une autre caractéristique intéressante est que ce code

compilé peut-être appelé depuis des pages HTML : on appelle ces programmes « applets ». Le code est téléchargé depuis des serveurs distants, par le protocole HTTP. L'inconvénient majeur de cette solution vient du temps de chargement de la première page HTML du site Web, qui implique le téléchargement de l'applet qui peut être assez volumineuse. Par exemple, dans le cas de HyperAtlas, elle est de 3 Mo par exemple.

Cependant, cette solution offre une API de programmation particulièrement riche avec une gamme étendue de fonctionnalités utiles en cartographie. Par exemple, pour le dessin, il existe les bibliothèques graphiques AWT et SWING ou bien Java2D[5] publiée par Sun[6]. Pour les opérations de topologie sur les formes vectorielles, une bibliothèque spécifique est disponible : *Java Topology Suite*[7] (JTS).

Ces technologies nécessitent l'installation de plugiciels (*plugins*) sur le poste client, pour interpréter ces formats qui ne sont pas lisibles directement pas les navigateurs. Flash utilise le lecteur fourni par la société Macromédia, qui est largement répandu, et pèse 400 Ko. Le format SVG est lu par le lecteur « SVG viewer » de Adobe, prédominant. Il pèse plus lourd que le *plugin* de Flash : 2,3 Mo. Mais il y a des alternatives, avec le nouveau lecteur de Corel, ou l'interprétation native présente dans le navigateur Mozilla. Pour les applets Java, il faut au préalable que le client soit équipé d'une machine virtuelle (JVM), pesant 10 Mo au chargement, ce qui n'est pas négligeable.
L'utilisation d'un *plugin* spécifique pose certains problèmes [@GeoClip]:

- son poids ne doit pas être trop important ;
- il doit être largement diffusé ;
- il doit être compatible avec tout navigateur.

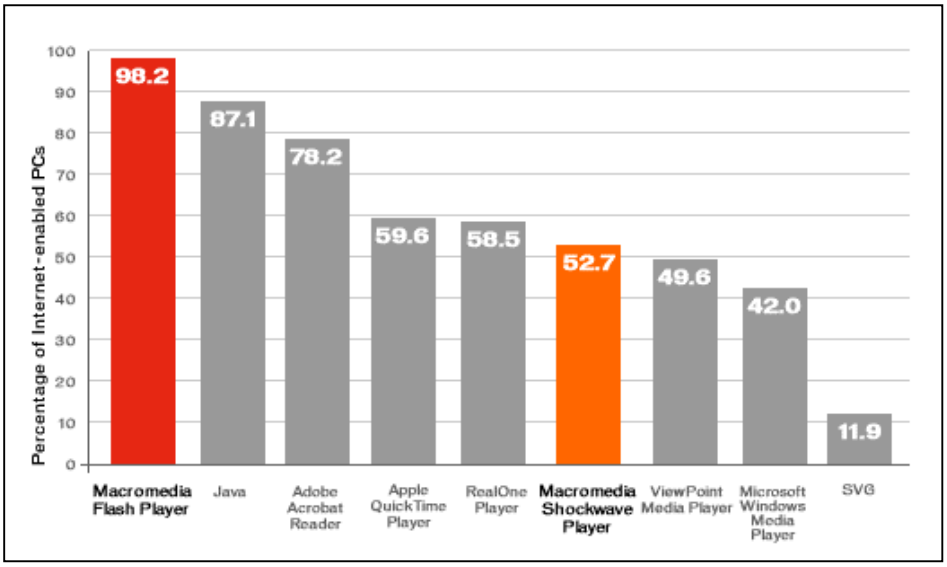


Figure 5. Statistiques d'équipement en plugin d'affichage. Source : NDP Research – médiametrix, décembre 2004

Ainsi, une enquête trimestrielle réalisée en Décembre 2004 auprès de 2000 personnes montre que le *plugin* le plus répandu est le « MacromédiaFlashPlayer », qui permet de lire le format Flash, tandis que le moins répandu est celui pour SVG : 11,9% seulement des postes équipés (cf. figure 5). Quant à la JVM, elle est déjà très répandue (en moyenne à 87%).

[5] http://java.sun.com/products/java-media/2D/samples/java2demo/Java2Demo.html
[6] Sun est la société qui détient la licence de Java : http://java.sun.com/
[7] http://www.vividsolutions.com/jts/jtshome.htm

- **Ajax**

Ajax est l'acronyme de *Asynchronous JavaScript And XML* (XML et Javascript asynchrones), et désigne une méthode informatique de développement d'applications Web. Ce n'est donc pas exactement un format de diffusion de données, mais plutôt la combinaison de technologies qui enrichissent le contenu HTML avec des instructions pour la manipulation dynamique des données :

- des feuilles de style CSS pour présenter l'information,
- JavaScript et le modèle objet DOM pour effectuer des calculs,
- l'objet XMLHttpRequest pour échanger les données de manière asynchrone avec le serveur Web,
- XML et XSLT pour la structuration des données.

Ce terme a d'abord été introduit par Jesse James Garrett, le 18 février 2005, dans un article sur le site Web *Adaptive Path* [@Ajax].

La nouveauté consiste à introduire le traitement par les navigateurs de l'objet *XMLHttpRequest*. Développé à l'origine par Microsoft en tant qu'objet ActiveX[8], puis intégré en tant qu'objet navigateur natif nommé *XMLHttpRequest* par Mozilla, il est aujourd'hui supporté par l'ensemble des navigateurs connus (*Internet Explorer 5.0*, *Mozilla 1.0*, *Safari 1.2*, *Opéra 8.0*, etc.). Cet objet est une recommandation du W3C depuis février 2007 [@W3C-XMLHttpRequest], et permet aux scripts de réaliser l'ensemble des opérations d'un client http : créer des requêtes HTTP, dont le type peut être précisé, envoyer des données et traiter leur résultat de façon asynchrone.

Cette approche allège le volume de données qui transitent entre le client et le serveur, car les données peuvent être acheminées par morceaux via l'emploi des objets *XMLHttpRequest*, en fonction des interactions de l'utilisateur sur la page que traite le script. Ainsi le serveur est aussi soulagé d'une partie des traitements sur les données (cf. figure 6). L'inconvénient de cette technologie réside essentiellement dans le temps de chargement de la première page qui peut-être assez long puisqu'elle inclue la bibliothèque Ajax (dont la taille peut être de 500 Ko par exemple).

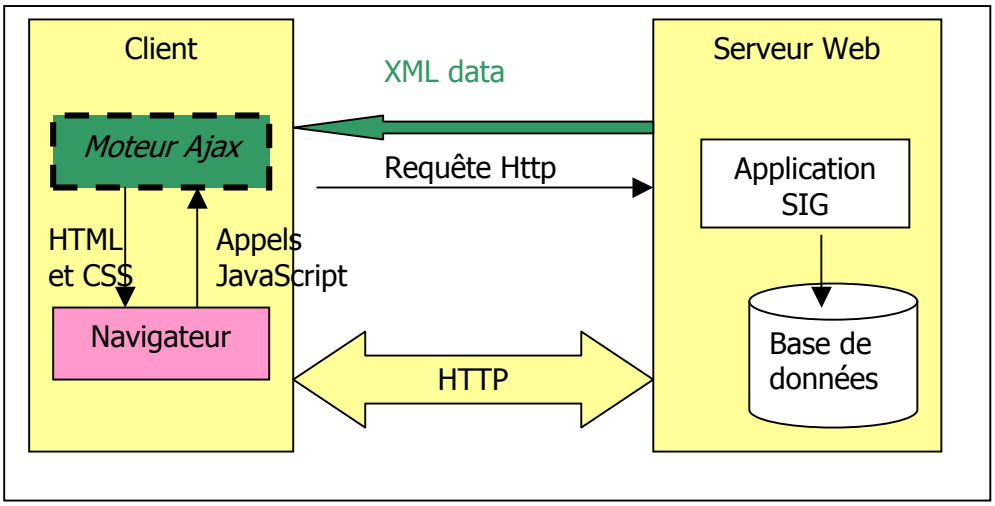


Figure 6. Architecture d'un service Web avec Ajax.

[8] Un objet qui est récupéré sur un serveur via Internet ou un réseau intranet, et contenant un code applicatif s'exécutant seulement sur des systèmes d'exploitation de Microsoft, mais qui accède alors à toutes les fonctions système (ce qui en fait un objet dangereux pour la sécurité des systèmes).

Avec ces quatre technologies, on peut déporter une part importante du traitement cartographique des données du côté du client. On peut ainsi programmer des comportements spécifiques en fonction des sollicitations de l'utilisateur : simple ou double clic de souris, sélection d'élément, agrandissement du cadre de l'image, etc. Ces solutions sont de plus indépendantes du système d'exploitation utilisé.

b) Pour une cartographie en 3D

Ce paragraphe cible les formats de données conçus pour programmer des animations et pour représenter des modèles 3D.

- **VRML et X3D**

Né en 1994, le *Virtual Reality Modelling Language* (VRML) est destiné à la représentation d'univers virtuels en 3D sur Internet, mais est utilisable pour la visualisation plane simple. Les différentes versions de VRML (VRML97, VXML1.0, VRML2.0) aboutissent à la spécification X3D, standard du Web3D consortium. [@Web3D]

Les programmes VRML sont des fichiers textes d'extension (.wrl). Ils se téléchargent depuis un serveur Internet et sont ensuite interprétés directement par le navigateur Web. Cependant, les navigateurs Web intègrent rarement cette fonctionnalité, et, dans ce cas, il faut installer un ***plugin* VRML** sur le client. La société Cortona édite un visionneur VRML[9] qui fonctionne avec Internet Explorer, Netscape, et Mozilla, sur Windows et Mac OS, et dont le poids est de 1.68 Mo, capable d'interpréter aussi GeoVRML. Le site GeoVRML[10] recommande aussi les deux *plugins* suivants : Cosmo Player 2.1.1 ou Netscape Live3D.

VRML inclut une description de la scène 3D et une programmation des évènements utilisateurs. La scène est décrite par des formes (sphères, cubes, cônes, cylindres...), des éclairages, des sons, des liens hypertextes, ainsi que leur agencement dans l'espace, leur texture, leur couleur, leur matériau. Le langage VRML définit une arborescence de nœuds, rassemblés dans des groupes. Les groupes s'utilisent pour définir des propriétés communes à des objets : repère géographique, réaction à un évènement particulier, etc. Les évènements sont gérés par un système de routage qui définit que faire en fonction des changements d'états de la scène ou des actions de l'utilisateur. Par exemple, cliquer sur une partie de carte déclenche un zoom sur la portion de carte.

- **MPEG-4**

MPEG-4 est un format de compression de données audio et vidéo [Bissler, 04] [Danzart et al., 2003]. Mais il permet également l'encodage de scènes composées d'éléments multimédia – son, vidéo, texte, graphique vectoriel 2D et 3D – synchronisés entre eux.

Norme conçue par le groupe de travail MPEG (Media Picture Expert Group) de l'ISO/IEC, en 1998, qui dérive de MPEG2, MPEG4 garde les avantages en termes de qualité et de compression du flux vidéo transmis avec MPEG2. Il introduit une hiérarchie arborescente d'objets décrivant les éléments d'une image. On ne parle plus d'images, mais de

[9] http://www.parallelgraphics.com/products/cortona/
[10] http://www.geovrml.org/

scènes audio-visuelles. La scène est constituée d'objets qui sont réutilisables et peuvent être encodés à des niveaux de détail différents. Ces objets peuvent être des photographies ou bien des images de synthèse, aussi bien en 2D qu'en 3D.

Pour décrire les scènes, MPEG4 utilise le langage BIFS (BInary Format For Scène). A noter que MPEG s'est inspiré du langage VRML pour créer BIFS. BIFS permet de coder des séquences d'animation 2D de façon aussi compacte qu'avec Flash. MPEG4 conserve aussi un des avantages de MPEG2 : le contrôle de flux en fonction du débit disponible. Ce contrôle s'effectue grâce à la hiérarchisation du codage en plusieurs couches, qui permet de sélectionner un niveau de qualité plus faible si les conditions de réception sont mauvaises.

L'utilisation d'une API Java permet de manipuler ces objets (en les déplaçant, en changeant l'angle de vue) ou, par exemple, d'intégrer des objets 3D dans une scène 2D. Elle permet surtout de fabriquer des scènes interactives dans lesquelles l'utilisateur peut :

- démarrer une séquence vidéo en cliquant sur un élément de l'image,
- changer de place un élément,
- changer son point de vue de la scène,
- changer les propriétés d'un objet (dimension, couleur…).

Il intègre aussi un support pour la gestion des droits numériques qui pourrait être très utile dans le cadre de la cartographie pour protéger l'accès à des cartes au contenu confidentiel (comme des cartes militaires). Cependant, tant que l'affichage de texte ne sera pas complètement normalisé dans MPEG4, ce standard restera inexploitable pour la cartographie.

2.1.3. Quelques infrastructures pour le support de données spatiales

Après avoir décrit quels formats et technologies peuvent être employées du côté du client, nous listons quelques infrastructures permettant de développer et d'exploiter un serveur de cartes.

a) A propos des standards

Nous exposons ici des standards pour l'échange et la communication des données, en dehors des formats de fichiers SIG. En effet, une organisation internationale fondée en 1994 et dédiée au développement des systèmes ouverts en géomatique, l'Open GIS Consortium[11] (OGC), a pour objectif de rendre les systèmes cartographiques interopérables. Son travail aboutit à des spécifications de langages, regroupées en fonction de l'objet de la communication. Nous n'en citons ici que 3, celles qui sont aujourd'hui le plus souvent implémentées, décrites par l'Infrastructure Canadienne de Données Géospatiales [@ICDG].

- **WMS**

La spécification visant les **services de cartographie Web (WMS)** définit un service permettant d'extraire une carte (ou une image) de données géoréférencées. Un système client est en droit de demander à un serveur implémentant le service WMS les opérations suivantes :

[11] http://www.opengeospatial.org/

- *GetCapabilities* retourne un fichier XML décrivant l'offre du serveur : sa version, les formats disponibles, les couches de données accessibles et les métadonnées qui décrivent la couche de données accessible ;
- *GetMap* retourne une image de carte dont les dimensions, le format, la méthode de projection, et la région géographique, les couches d'informations correspondent à celles spécifiées par le client ;
- *GetFeatureInfo* retourne de l'information sur des entités particulières représentées sur une carte. GetFeatureInfo est facultatif.

- **WFS**

La spécification visant les **services de fonctionnalités Web (WFS)** définit un ensemble d'opérations visant l'extraction et la manipulation d'entités géographiques. La spécification permet deux niveaux de fonctionnalité. Un service WFS de base ne permet que l'extraction d'entités, tandis qu'un service WFS de transaction offre en outre la possibilité de manipuler des entités.
Les opérations offertes par un service WFS de base sont :

- *GetCapabilities* qui retourne les métadonnées décrivant les types d'entités qui sont disponibles sur le serveur ;
- *DescribeFeatureType* qui retourne le schéma d'un ou plusieurs types d'entités demandés ;
- *GetFeature* qui retourne les entités correspondant à une requête spécifique.

Les services supplémentaires offerts par un service WFS de transaction sont :

- *Transaction* qui permet de créer, mettre à jour ou supprimer des entités ;
- *LockFeature* qui verrouille les entités pour empêcher que des tentatives de modification d'une même entité soient effectuées simultanément.

En option, les requêtes *GetFeature*, *Transaction* et *LockFeature* peuvent inclure un « filtre » sélectionnant les entités sur lesquelles les opérations portent. La spécification «*Filter Encoding Implementation Specification*» détermine un format pour ce filtre.

La spécification visant un service WFS précise que les entités doivent être échangées au moyen du *Geographic Markup Langage* (GML), littéralement langage de balisage géographique**.**

- **GML**

Il fournit une grammaire pour l'encodage du contenu géospatial (propriétés, emplacement, étendue des entités, relations entre des entités) des entités. Le schéma sur lequel se base le langage GML offre un ensemble d'éléments XML normalisés pour décrire des entités spatiales et des types géométriques.

Notamment, ce langage précise le système de projection utilisé pour calculer les coordonnées d'un lieu, ce qui permet d'intégrer différents référentiels de données dans un même document. Ce format caractérise donc le monde réel en utilisant des attributs géométriques et topographiques. Par, contre il ne permet pas de décrire le monde d'un point de vue sémantique (pas de distinction entre pays, région, ville, quartier, etc.).

D'un point de vue développeur, ce langage a l'avantage d'être supporté par Java qui propose une API pour concevoir des documents GML avec la librairie GML4J[12].

[12] http://sourceforge.net/projects/gml4j

GML s'utilise pour exporter des données géographiques : il ne dit pas comment les représenter, mais il les décrit de façon normalisée. Il permet donc un déport partiel des données pour réaliser des traitements locaux sur un client. Et c'est aussi un moyen pour mettre en œuvre l'interopérabilité entre des sites Web cartographiques. C'est la raison pour laquelle la spécification WFS recommande son utilisation. Sur le site de l'infrastructure canadienne de données géospatiales [@ICDG], on trouve un exemple complet d'utilisation de GML dans le cadre d'un service WFS.

b) Quelques SIG

- **Les SIG commerciaux**

Il existe plusieurs concepteurs de logiciels SIG commerciaux (APIC[13], ESRI[14], GéoConcept[15], GéoMédia[16], MapInfo[17], etc.). Ces éditeurs déclinent généralement leurs produits en quatre grandes catégories d'outils :

- les visionneuses qui correspondent à des versions allégées du logiciel SIG et qui permettent une première approche de l'information géographique par la visualisation des données. Ces outils sont gratuits et d'une utilisation très simple (Mapinfo Proviewer, ArcExplorer, etc.).
- les logiciels stricto sensu qui offrent la gamme complète des fonctionnalités spécifiques aux SIG depuis la saisie et l'analyse des données jusqu'à la restitution cartographique (MapInfo professionnel, ArcGis, GéoConcept expert, etc.). Ces logiciels existent en version mono-poste pour un coût de 3000 € environ ou en version serveur avec des systèmes de licences fixes (plusieurs utilisateurs simultanés en nombre connu) ou flottantes (plusieurs utilisateurs simultanés en nombre variable). Le premier prix d'une solution serveur est d'environ 12 000 €. Ce tarif de base peut être multiplié par 3 voire 4 en fonction du nombre de postes clients, du nombre de requêtes autorisées, du contrat de maintenance, etc. Les logiciels SIG sont accompagnés par un langage de programmation qui permet de compléter leurs fonctionnalités. Ils peuvent également s'enrichir de modules complémentaires (visualisation 3 D par exemple) dont les coûts varient de 1000 à 3000 €.
- les applications métiers développées autour d'un noyau SIG avec les langages de programmation cités précédemment. Ils visent à présenter à l'utilisateur des fonctions et une interface adaptées à son métier et non l'interface standard du SIG. Le coût de ces solutions varie de 200 à 2500 € selon les thématiques et les outils utilisés.
- enfin, il existe des SIG destinés à être utilisés via Internet. Ils présentent l'avantage de limiter le nombre de licences car leurs fonctionnalités sont accessibles grâce à un navigateur Internet standard.

[13] http://www.star-apic.com/
[14] http://www.esri.com
[15] http://www.geoconcept.com/fr/
[16] http://www.geoconcept.com/fr/
[17] http://www.mapinfo.com/

Ces SIG proposent des fonctionnalités intéressantes, et de plus, les sociétés les commercialisant se sont attachées à autoriser l'emploi du format de données des concurrents (MIF/MID pour MapInfo, Shapefile (.shp) pour ESRI, etc.). Le principal problème vient du coût de leur licence d'exploitation, assez élevé.

- **SIG libres**

La démarche Open Source s'étend aux systèmes d'information géographique (SIG), très utilisés, notamment, dans les collectivités territoriales. L'Open Source Geospatial Foundation (http://www.osgeo.org/) oeuvre au développement collaboratif des données et des technologies géospatiales ouvertes. Les SIG libres (sous licence LGPL) sont de plus en plus exploités par l'industrie cartographique [@LeMondeInformatique], qui développe des applications spécifiques pour le compte de clients. CampToCamp[18], par exemple, est à l'origine d'un module de génération de cartes thématiques exploitant les technologies Ajax, réalisé pour l'Ifremer. Géomatika[19], de son côté, met en place des serveurs cartographiques (IsiGéo collectivités par exemple) basés sur le noyau Mapserver.

Le tableau 1 liste les SIG libres les plus connus, synthétisant leurs capacités, et la technologie qu'ils emploient.

GIS	Technologie	Formats d'entrée et sortie	OS	Base de données
MapServer http://mapserver.gis.umn.edu/	Ecrit en C, propose un wrapper MapScript pour PHP, Perl, Pyton, Java	Vecteur : SVG, Flash Matriciel : oui Fichiers SIG : MIF/MID, .shp Services Web : WMS, WFS	Unix/ Linux/ Windows	Oracle, Sybase, MySQL PostgreSQL
GeoServer http://docs.codehaus.org/display/GEOS/Home	Java	Vecteur : GML, SVG Matriciel : JPEG, PNG, TIFF Fichiers SIG : .shp Service Web : WMS, WFS avec transaction.	Tous avec JVM	PostgreSQL, PostGIS Oracle
OpenMap http://openmap.bbn.com/	Java	Vecteur: oui Matriciel : oui Fichiers SIG: .shp		Oracle
OpenJump http://openjump.org/wiki/show/HomePage	Java	Vecteur : oui Matriciel: oui FichiersSIG: .shp, Services Web : WMS, WFS		Oracle, PostgresSQL, PostGIS
Deegree http://www.deegree.org/	Java	Vecteur : GML Matriciel : JPEG, GIF, PNG, TIFF, et BMP. Fichiers SIG : .shp, Services Web : WMS, WFS		Oracle, PostgreSQL, MySQL
GRASS http://grass.itc.it/	C/C++ et bibliothèque OGR	Vecteur : VRML Matriciel : TIFF, GIF, PNG, ASCII Fichiers SIG : .shp, MIF/MID	Unix/ Linux/ MacOS	PostgreSQL, Oracle
QuantumGIS http://qgis.org/	C++, Qt et bibliothèque OGR	Vecteur : SVG Matriciel : JPEG, TIFF, PNG Fichiers SIG : MIF/MID, .shp Service Web : WMS	Unix/ Linux/ Windows/ MacOS	PostgreSQL

Tableau 1. Liste de SIG sous licence libre.

[18] http://www.camptocamp.com/
[19] http://www.geomatika.fr/

2.2. Les formes de représentation en analyse spatiale

Après avoir répertorié les outils pour la création et la diffusion de cartes, nous présentons dans cette partie leur contenu. En particulier, quelles sont les formes de représentation cartographiques disponibles, leurs avantages, leurs inconvénients.

2.2.1. Une représentation plane

Selon les règles de la sémiologie graphique de J. Bertin [Bertin, 1974], la représentation correcte d'une variable résultant d'un comptage doit employer un symbole de taille proportionnelle (comme l'aide d'un disque), tandis que les variables qui ne sont pas additives doivent utiliser une représentation basée sur l'intensité d'un paramètre (comme la couleur).

Les cartes en **cercles proportionnels** sont donc destinées à représenter des quantités ou des effectifs : l'aire des disques est proportionnelle à la valeur de la variable quantitative. On dessine un cercle par unité, dont le centre peut avoir une raison géographique (chef-lieu, commune, etc.), ou simplement géométrique (centroïde).

Cependant, l'emploi de ce type de représentation nécessite quelques précautions : la surface totale des disques ne doit pas excéder 10% des terres émergées visualisées. Ce qui nécessite donc un calibrage soigneux de la taille des cercles. D'autre part, dans le cas où la densité des unités est élevée, les disques peuvent se chevaucher ou couvrir la surface entière des unités, rendant illisible la carte. Une solution pour le chevauchement est le détourage (une circonférence tracée en blanc) permettant de distinguer les petits cercles placés sur les grands. Ce mode de représentation n'est donc véritablement efficace que lorsque les unités spatiales ne sont pas en trop grand nombre (quelques centaines au maximum), et qu'elles sont réparties dans l'espace de manière relativement homogène.

Les cartes en cercles proportionnels peuvent donner lieu à diverses variations qui permettent d'en enrichir le contenu, sans pour autant en lever les limitations.

Les cartes en **cercles proportionnels colorés** permettent de représenter simultanément des effectifs et des valeurs non additives (comme un ratio, un pourcentage, etc.) Dans ce cas, la taille du cercle devient proportionnelle à la variable quantitative, tandis que sa couleur dépend de la valeur non additive. Par exemple, elle s'emploie pour illustrer la vétusté des logements en France en montrant la proportion de résidences principales équipées de WC extérieurs dans les départements français en 1991, cf. figure 7.

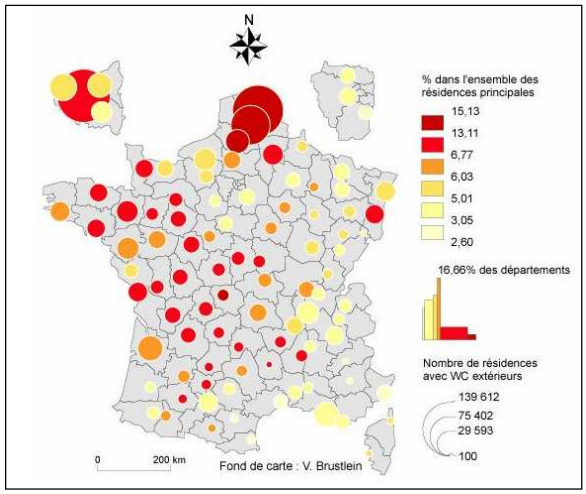


Figure 7. Nombre de résidences principales avec WC extérieurs en 1991 par départements français. [@Philcarto]

Les cartes en **demi-cercles affrontés ou cocardes** permettent aussi de comparer deux variables, en accolant deux demi-cercles pour chacune de couleur différente, donnant ainsi une lecture directe de la corrélation qui peut exister entre leurs surfaces respectives.

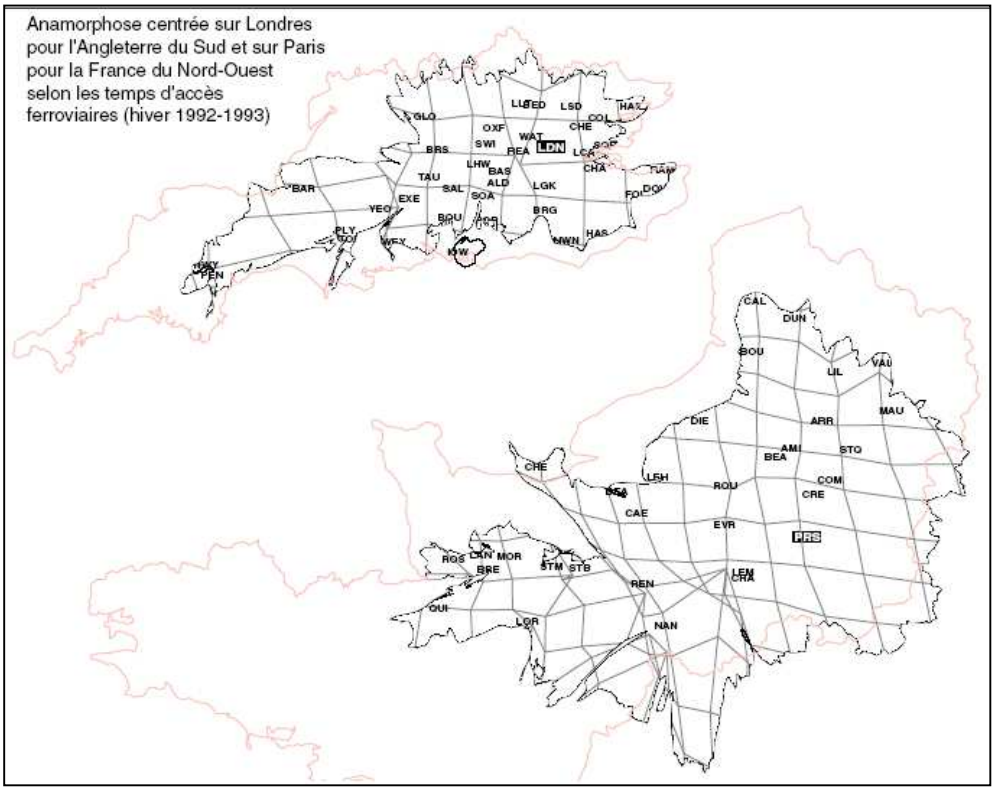
La **carte choroplèthe** est le type le plus usuel de carte statistique. Il s'agit d'une représentation de **quantités relatives** à des espaces, ou aires géographiques, par le moyen d'une échelle de tons gradués. Par exemple, HyperAtlas utilise des cartes de ce type pour représenter les ratios d'indicateurs et leurs déviations.

Ce type de carte pose un problème à la fois mathématique et graphique. Sa réalisation repose d'abord sur le choix d'une méthode de discrétisation, c'est-à-dire de division de la série statistique que l'on veut cartographier en classes, ou intervalles. Il faut par ailleurs déterminer le nombre de classes retenues pour la représentation, qui correspond au nombre de paliers visuels à réaliser. Le nombre maximum de paliers de valeur (donc de classes de données) oscille par exemple entre 5 et 10, selon les auteurs. Il existe plusieurs dizaines de méthodes de discrétisation, qui peuvent être réparties en quelques catégories : méthodes mathématiques, reposant sur les valeurs des données, méthodes statistiques ou probabilistes, reposant sur les fréquences, méthodes graphiques, demandant la construction de diagrammes ou courbes auxiliaires. La plupart des logiciels de cartographie permettent de réaliser des représentations suivant les principales méthodes de base (classes d'égale étendue, classes standardisées, etc.) et génèrent automatiquement des séries de valeurs échelonnées. Le choix des palettes de couleurs a aussi son importance, et fait l'objet d'études spécifiques [Brewer, 2004].

2.2.2. Une vision déformée

Des déformations de plusieurs types peuvent être introduites sur le maillage initial afin que la forme et la surface des unités résultant de cette transformation soient représentatives d'une variable mesurée. Colette Cauvin introduit une typologie complète pour ces déformations [Cauvin, 1998], mais nous n'en décrirons que deux : les transformations thématiques de poids (les cartogrammes en sont un exemple) et les transformations thématiques de liens avec les anamorphoses par exemple.

Une **anamorphose géographique** est un procédé cartographique qui consiste à transposer, à transcrire une variable descriptive des lieux en une variable déformant ces lieux. En ce sens, c'est une métaphore spatiale [Denain et al., 1998]. Elle peut s'utiliser par exemple pour décrire les relations de proximité géographique entre lieux, fondées sur des distances temps. La déformation de l'image se calcule en appliquant un vecteur de déformation en chaque point, proportionnel à la variable que l'on désire représentée. C'est une anamorphose vectorielle.



L'exemple le plus connu compare l'accessibilité depuis Londres et Paris des villes provinciales anglaises et françaises, (cf. figure 8).

Figure 8. Exemple d'anamorphose vectorielle

Un cas particulier d'anamorphose est le cartogramme : dans ce cas, la déformation des unités d'aire est proportionnelle à la valeur d'un indicateur résultant de comptage. Ce type de représentation convient pour représenter des ratios dont le dénominateur n'est pas proportionnel à la taille des unités, comme par exemple le PNB par habitant. Cette représentation donne une perception directe à l'utilisateur de l'importance de l'unité observée, relativement à la quantité étudiée (nombre d'habitants, etc.). L'exemple de la figure 9 compare une carte classique (à gauche) des états américains colorisés en fonction du vote majoritaire (rouge pour Bush, bleu pour Kerry) aux élections présidentielles américaines en 2004 avec une carte où les états sont transformés relativement à leur population (à droite). Il a pour but d'expliquer que le système électoral américain, qui repose sur une hypothèse fausse d'équirépartition de la population dans les états, peut faire élire un président ayant obtenu moins de voix que son concurrent. [@Elections2004-anamorphose]

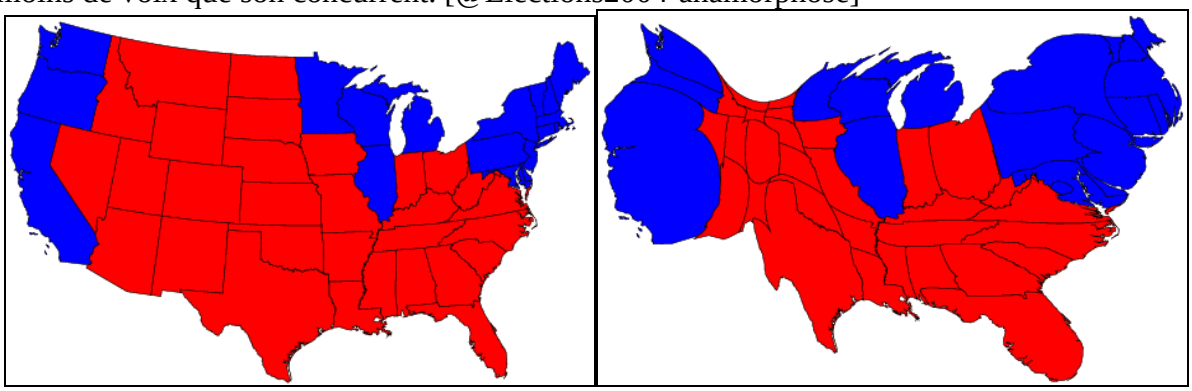
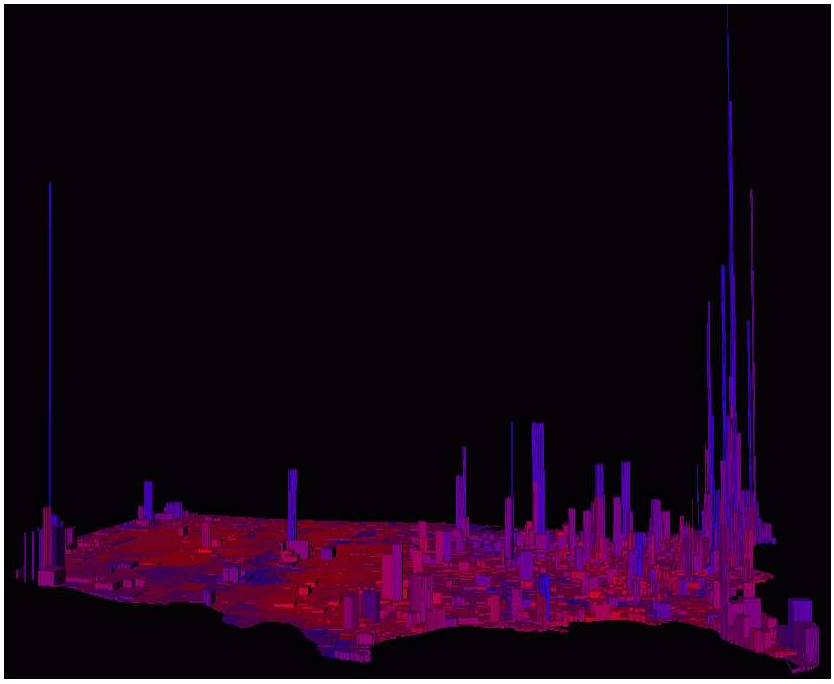


Figure 9. Utilisation de cartogramme pour l'analyse de la répartition du vote républicain et démocrate des élections présidentielles américaines de 2004.

2.2.3. Une vision 3D

La cartographie 3D est aussi un atout pour la communication de l'information. La troisième variable z introduite peut avoir deux significations différentes : elle peut représenter l'altitude d'un terrain au sens topographique, ou bien la valeur d'une variable thématique. Cette seconde signification nous intéresse pour l'analyse spatiale, car alors une représentation 3D permet d'ajouter des données abstraites aux paysages.



Par exemple, la figure 10 illustre que les zones les plus urbaines et peuplées des Etats-Unis ont majoritairement voté Démocrate lors des élections présidentielles 2004 des Etats-Unis.[@Elections2004-3D] La variable z représente le nombre de votants par unité d'aire (le mile). Le bleu indique un vote majoritaire en faveur de Kerry, le rouge un vote majoritaire pour Bush. Cette image est une capture d'écran d'un modèle 3D, dans lequel on peut zoomer, et se déplacer.

Figure 10. Modèle 3D figurant les résultats des élections présidentielles américaines 2004.

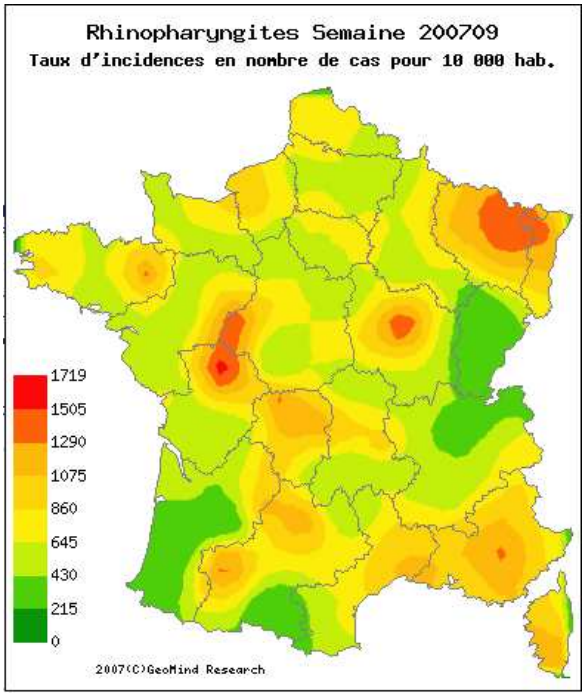
2.2.4. Une représentation continue

Curieusement, on parle de représentation continue lorsque la carte vise à faire émerger des zones de disparités (ou discontinuités) qui ne suivent pas les frontières administratives habituelles. Ces représentations mettent en jeu diverses méthodes d'interpolation : méthode des moindres carrés, lissage par voisinage, Krigeage, méthode de Shepard, etc., qui sont présentées en annexe de ce document. Nous présentons un aperçu des diverses formes de représentation auxquelles elles peuvent donner lieu. Ces cartes synthétisent l'organisation générale du phénomène étudié en réduisant l'importance des aspérités d'une distribution statistique.

a) Les lignes et les surfaces de tendances

Il s'agit ici d'associer des lignes ou bien des surfaces de même niveau à une variable Z que l'on veut représenter, et de découvrir la forme de la fonction mathématique f associant les coordonnées géographiques (X,Y) à Z : f(X,Y) = Z. L'inconvénient majeur peut venir du coût en temps de calcul de la résolution de cette équation, lorsque Z ne peut être le résultat d'une mesure directe.

La représentation par **courbes de niveau** du relief, auxquelles les cartes de randonnée nous ont habitué, et où les points de même altitude sont reliés entre eux par une ligne, se généralise aisément et s'applique à toute sortes de variables géographiques : la pluviosité (isohyètes), la température (isothermes), etc. L'intérêt d'une telle représentation est de visualiser aisément le gradient de la variable étudiée : les lignes se resserrent lorsque le phénomène évolue fortement sur une zone. On peut moduler la densité des lignes en restreignant le nombre de niveaux représentés.

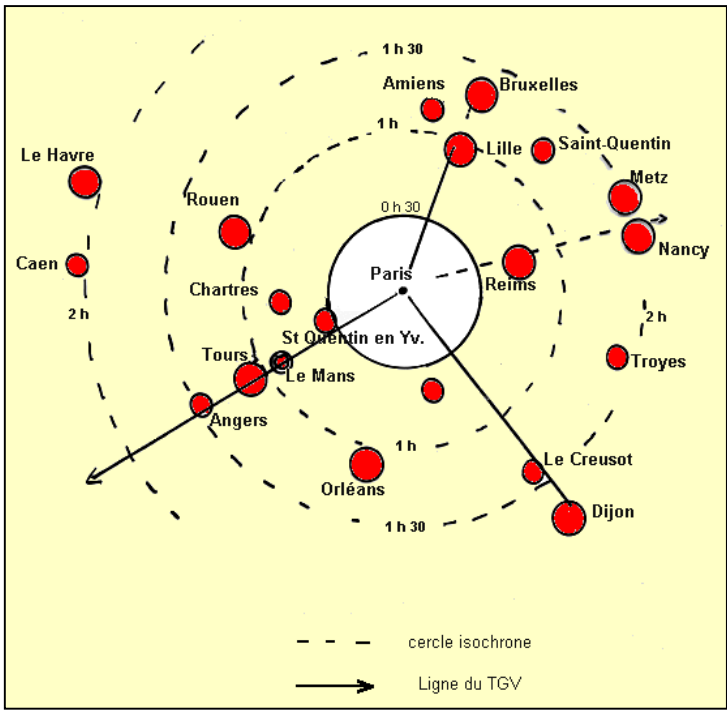


Elles s'appliquent par exemple à l'étude et la représentation de phénomènes épidémiologiques. M.Toubiana propose en ligne sur un site Web[20] une cartographie par isolignes des épidémies hivernales en France, mettant en évidence les foyers infectieux, et la diffusion spatiale des maladies (cf. figure 11), fondée sur l'emploi de la méthode du Krigeage.

Figure 11. Cartographie d'une maladie basée sur l'emploi de la méthode du Krigeage - Source : http://bounty.necker.fr/hivernale/, 18 Mars 2007

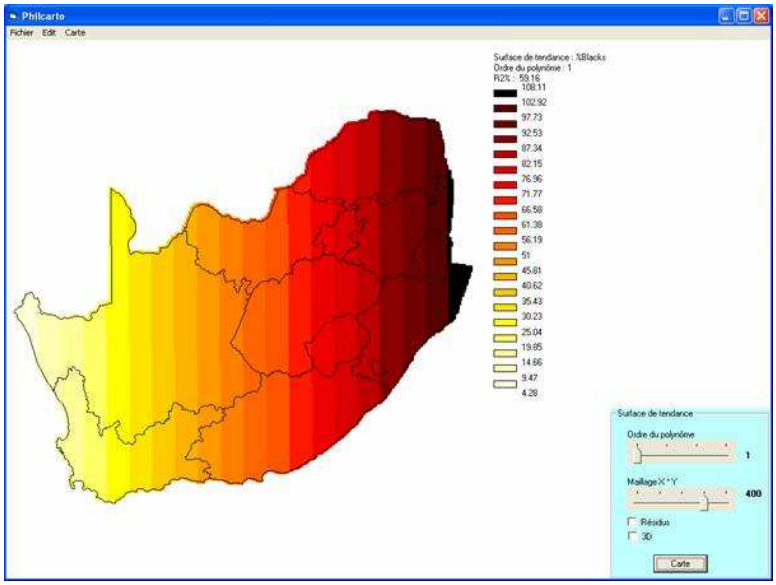
Un cas d'application intéressant concerne la représentation par isochrone : ce type de carte matérialise les temps de parcours (en transport collectif ou autre) à partir d'un lieu donné et permet d'appréhender l'accessibilité des territoires, cf. figure 12.

Figure 12. Le transport ferroviaire à grande vitesse : position des villes du bassin parisien en fonction des cercles isochrones. [@RennesAC]



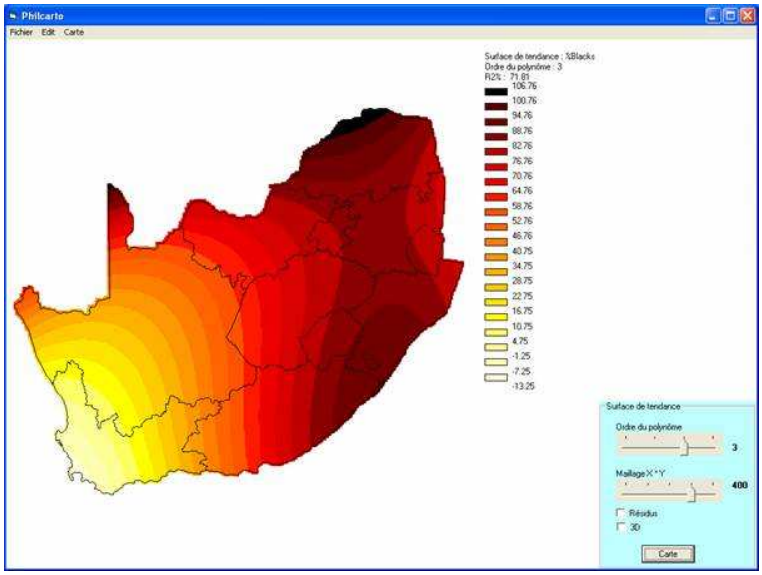
[20] http://bounty.necker.fr/hivernale/

L'analyse en **surfaces de tendances** utilise la **méthode des moindres carrés** pour ajuster des polynômes bidimensionnels *:* $z_i = f(x_i, y_i)^P$ où P représente l'ordre du polynôme. On obtient ainsi une série de surfaces de plus en plus complexes quand s'accroît l'ordre du polynôme. Pour représenter les surfaces tridimensionnelles sur le plan cartographique, on recourt à des plages de niveaux colorées ; la gamme bicolore permet d'étendre à 16 le nombre de plages, ce qui améliore la lisibilité.



Cette technique peut s'appliquer à la carte de la population de race noire dans les districts sud-africains en 1991. On détecte deux tendances très significatives : la première correspond à un ajustement par un polynôme d'ordre 1 qui absorbe 58% de la variance totale (cf. figure 13). Cette tendance, qui correspond à une surface plane, exprime l'accroissement de la part de la population de race noire dans la population totale d'Ouest en Est.

Figure 13. Pourcentage de population de race noire dans la population totale en Afrique du Sud, 1991. Surface de tendance d'ordre 1 (R^2 = 58%).



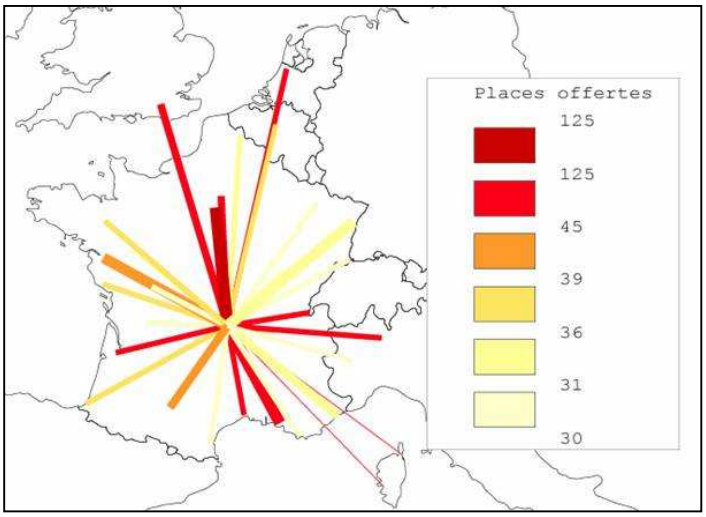
En ajustant un polynôme d'ordre 3, le pourcentage de variance expliquée atteint 72%. Cette tendance, plus complexe que la précédente, prend en compte l'existence d'une région exclusivement noire, le Transkei situé au Sud-Est du pays, et le mélange racial tout relatif de la province de Johannesbourg. D'où cette forme resserrée des plages de niveau sur la partie Est du pays sur la figure 14.

Figure 14. Pourcentage de population de race noire dans la population totale en Afrique du Sud, 1991. Surface de tendance d'ordre 3 (R2 = 72%).

b) Les cartes de liens

Les cartes en **oursins** permettent de visualiser les aires d'influences des lieux centraux en fonction des liens qu'ils entretiennent avec les niveaux de la hiérarchie urbaine qui leur sont immédiatement inférieurs et supérieurs. Les lignes reliant les points deux à deux peuvent être coloriées en fonction des valeurs prises par une variable. Ce mode de représentation est extrêmement efficace lorsqu'on cherche à exprimer une organisation spatiale polarisée car les liens s'arrangent alors à la manière des piquants sur un oursin.

Les flux qui sont échangés d'un point à un autre peuvent être inégaux. Dans ce cas, il est intéressant de rendre les épaisseurs des liens proportionnelles aux quantités ou aux effectifs représentant les mesures de ces flux. On peut aussi représenter simultanément des quantités ou des effectifs (épaisseurs des liens) et des valeurs numériques continues ou discrètes exprimant des pourcentages, des mesures, etc. (couleurs des liens).



Frédéric Dobruszkes.

Par exemple, le nombre de fréquences aériennes au départ de Clermont-Ferrand peut être complété par le nombre moyen de sièges offerts par vol ce qui permet d'apprécier la hiérarchie des relations de manière plus précise en prenant en compte non seulement la fréquence des vols, mais aussi la capacité des avions assurant telle ou telle autre liaison, comme l'illustre la figure 15.

Figure 15. Fréquences aériennes au départ de Clermont-Ferrand et nombre de sièges par vol. Fond de carte et données :

c) Les cartes issues de carroyage

Le **carroyage** est une méthode de rassemblement et de traitement de données (aux propriétés additives) en vue d'une exploration statistique et cartographique consistant à diviser l'étendue en carreaux (ou bien en polygones réguliers) égaux et repérés. Le carroyage a comme principal intérêt de permettre des analyses fines à des échelles locales, tout en préservant l'anonymat des personnes. En effet, la loi française[21] interdit l'exploitation d'îlots statistiques peuplés de moins de 5000 habitants, pour protéger leur vie privée. [Bizet, 1998].
Un cas d'usage fréquent concerne les données de santé, considérées comme très sensibles en France [Bussi, 2000]. Le carroyage peut aussi être vu comme une solution au problème de modification des maillages dans le temps [Grasland, 2006].

[21] C'est une recommandation issue de la Commission Nationale Informatique et Libertés (CNIL)

Techniquement, le carroyage peut :

- Supposer une équirépartition de la variable dans les zonages statistiques, et répartir sur chaque cellule la valeur statistique étudiée proportionnellement à la surface d'intersection entre cellule et taille de la maille du zonage.
- Supposer au contraire une hétérogénéité de densité à l'intérieur des mailles du zonage (cas le plus réaliste). Dans ce cas, on peut alors croiser une base cadastrale par exemple pour déterminer l'ensemble des cellules recevant une part de la redistribution de la variable.

Par contre, les cartes carroyées ne sont pas forcément de lecture aisée : pour améliorer ce point, un **lissage** (par moyenne mobile par exemple) peut-être effectué sur la mosaïque obtenue (cf. figure 16).

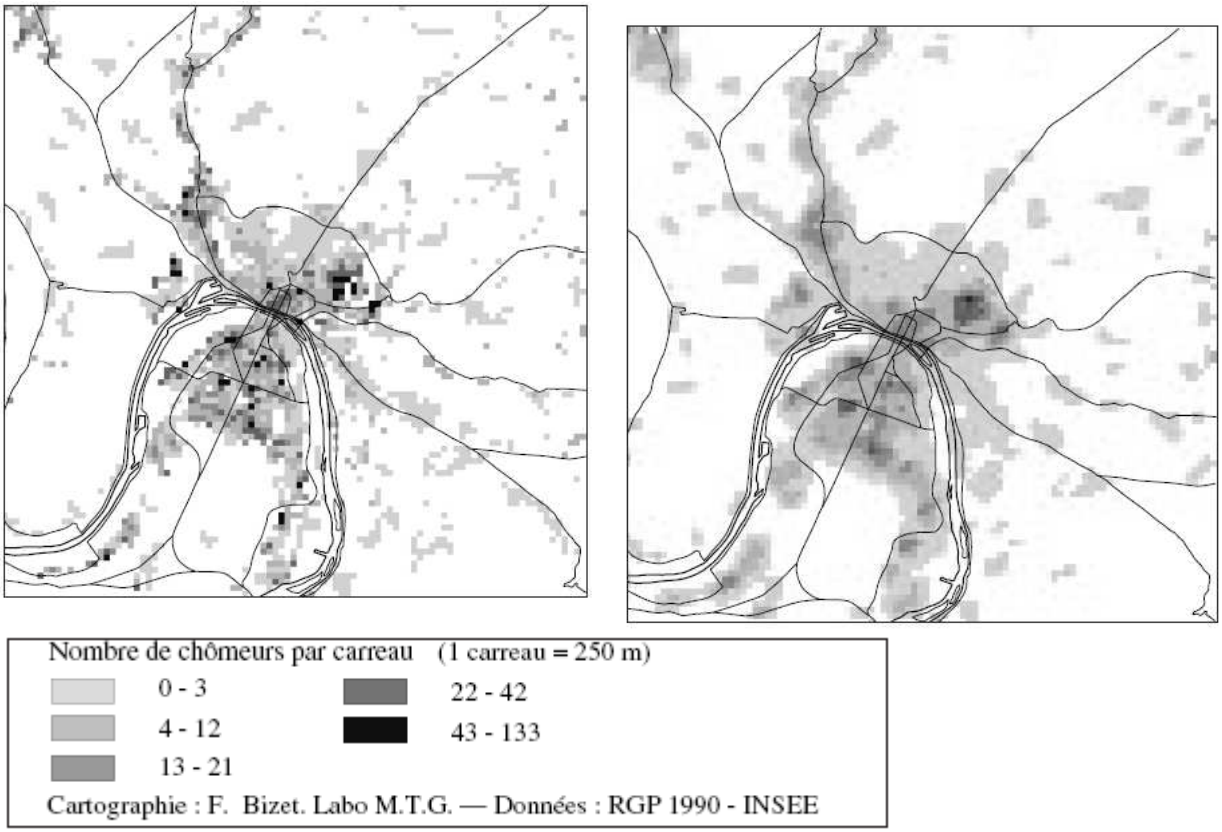


Figure 16. Carte de répartition des chômeurs depuis plus d'un an sur Rouen en 1990. A droite, avec lissage du carroyage par moyenne mobile.

2.3. Synthèse

En conclusion, il existe un panel de technologies et d'outils très large pour faire de la cartographie, mais combien sont directement liés à l'analyse spatiale multi-scalaire, et territoriale ? Nous tentons de répondre à cette question dans cette synthèse, en indiquant les technologies utilisées, et les formes de représentation qu'ils proposent. Sans entrer dans le détail de leurs fonctionnalités, on peut aussi se demander s'ils offrent une certaine

interopérabilité avec d'autres outils, et un moyen pour administrer les données qu'ils exploitent.

Les principaux outils connus de cartographie interactive, offrant des fonctions d'analyse spatiale sur des statistiques socio-économiques sont : GéoClip[22], le Cartographeur[23], Philcarto[24], l'AtelierC@rto[25], SCAP-3[26] et WinCarto[27].
Ces logiciels sont commercialisés par des sociétés privées (cas de GéoClip et du Cartographeur) ou bien il sont conçus et distribués par des enseignants-chercheurs, ou des collectivités territoriales (PhilCarto, l'AtelierC@rto, SCAP-3 et WinCarto). Dans le second cas, l'objectif principal est de mettre à disposition du grand public un outil pédagogique pour faire de la cartographie thématique.

Ces outils sont parfois très riches en terme de fonctionnalités. Le tableau 2 offre une vue synthétique des fonctionnalités, technologies employées par ces logiciels.

Toutefois, nous apprécions aussi les qualités ergonomiques des logiciels, en rapport avec notre objectif de cartographie grand public. Par exemple, l'utilisation du Cartographeur n'est pas simple. En outre, le respect ou non des règles de sémiologie graphique est aussi un critère de qualité. Ainsi l'AtelierC@rto ne respecte pas forcément très bien les normes de sémiologie : il est ainsi possible de produire une carte à cercles proportionnels trop nombreux et denses, qui rendent la carte illisible.

[22] http://www.geoclip.net/fr/index.php
[23] http://www.cartesetdonnees.com/FR05/EVA/cartoUniversel.htm
[24] http://philgeo.club.fr/Index.html
[25] http://www.iaat.org/Offre/CartoInteractive.php?id2=14&rub=util&onglet=presentation
[26] http://w3.univ-tlse2.fr/geoprdc/scap/index.php?page=presentation
[27] http://soshg.free.fr/wincarto/index.htm

Logiciel	Technologie	Formes de représentation			
		Plane	Déformations	3D	Continue
GéoClip	Flash	Cercles proportionnels Choroplèthes Oursins	non	non	Lissage par moyenne mobile
Cartographeur	Inconnue	Cercles proportionnels Choroplèthes Histogrammes Vecteurs de flux Oursins	Cartogramme	oui	Carroyage Polarisation Isolignes
PhilCarto	Ajax	Cercles proportionnels Choroplèthes Vecteurs de flux Oursins	non	oui	Isolignes, Surfaces de tendance Carroyage
AtelierC@rto	Ajax	Cercles proportionnels Choroplèthes	non	non	non
SCAP-3	Java	Cercles proportionnels Choroplèthes	non	non	non
WinCarto	C++ (utilisation des MFC[28])	Cercles proportionnels Choroplèthes Histogrammes	non	non	non

Tableau 2. Comparaison de logiciels d'analyse spatiale territoriale similaires à *HyperAtlas*.

En ce qui concerne l'interopérabilité, on constate qu'aucun des logiciels cités ne proposent une implémentation des services WMS ni WFS, ou bien d'export ou import de données dans le format GML.

Par ailleurs, les possibilités d'intégration et d'acquisition de données offertes par ces différents logiciels sont très inégales :

- En ce qui concerne l'accès et l'interrogation de bases de données, GéoClip dans sa version serveur est prévue pour importer des données extraites de SGBDR (*SQL Server*, *PostGre SQL*, *Oracle*, *MySQL*). En version client léger, il autorise simplement l'import de fonds géométriques au format MIF/MID ou bien *shapefile*. Dans les deux cas, les séries statistiques peuvent être lues à partir de fichiers de type tableur (au format Excel) sélectionnés depuis l'interface graphique. Pour l'export, il n'est prévu aucun autre format que les fichiers .swf (de Flash). Et d'autre part, au delà de 6000 objets, GéoClip ne peut montrer la carte dans sa totalité, et visualise la carte par portions. Le déplacement en dehors de l'une de ces portions entraîne le rechargement des données et un nouveau tracé de la carte.
- Le Cartographeur est lui capable d'importer des données dans divers formats de SIG connus (shapefile, MIF/MID ou AdobeIllustrator 8, par exemple), et il permet ensuite d'exporter les cartes produites aux formats suivants : PNG, GIF, JPEG, PDF, dans des pages HTML, Descriptions Postscript (.ps), Windows95 metafile (.emf), et dans Flash (.swf). D'autre part, les données de séries statistiques (les stocks) peuvent être importées via une interface graphique à partir de données stockées dans des feuilles Excel. Mais, les données une fois importées ne peuvent

[28] Microsoft Foundation Classes, bibliothèque de fonctions graphiques s'interfaçant uniquement avec le système d'exploitation Windows de Microsoft.

pas être modifiées. Les modifications doivent être effectuées sur les fichiers de données avant import.

- PhilCarto permet de créer et charger ses propres jeux de données (fonds de carte au format ouvert d'AdobeIllustrator et séries statistiques provenant du tableur Excel). Il exporte les cartes produites au format AdobeIllustrator.
- L'AtelierCarto est prévu pour s'interfacer avec des bases de données de niveau régional en entrée, par un accès protégé. Pour l'export, les données sont sorties au format PDF ou bien dans des fichiers Excel.
- SCAP-3 travaille avec des formats de fichier spécifiques, mais il est capable d'importer les données statistiques au format Microsoft Excel et les fonds de cartes polygonaux au format MIF/MID.
- WinCarto est capable d'importer dynamiquement, via l'interface graphique, des fonds de cartes au format PCX, ainsi que des séries statistiques provenant de tableurs au format SLK. L'inconvénient est que ce sont des formats spécifiques pour des logiciels fonctionnant sous le système d'exploitation Windows. Il exporte les cartes dans des formats matriciels : BMP, GIF, JPG ou PCX.

Pour conclure, cet état de l'art nous amène au constat suivant : la technologie Java, qui offre beaucoup de possibilités en matière de cartographie interactive, est déjà largement employée dans les SIG. Par ailleurs, les outils similaires à *HyperAtlas* proposent aujourd'hui des moyens pour l'acquisition de données : il semble que le format d'import pour les données le plus répandu soit des feuilles Excel pour les données statistiques, et que pour l'import de fonds de carte, ce soit MIF/MID. Les plus avancés (PhilCarto, le Cartographeur) offrent aussi des formes de représentation continue, qui affranchissent le lecteur des maillages administratifs.

3. Analyse de l'existant et proposition

Dans ce chapitre, nous décrivons les besoins actuels du projet Hypercarte, et afin de comprendre notre proposition composée de deux nouveaux modules, nous livrons une analyse technique du module existant, HyperAtlas.

3.1. Présentation des nouveaux besoins

Nous présentons ici les désirs exprimés par les membres du projet en ce qui concerne des idées scientifiques ou bien des besoins pratiques, et qui ne concernent pas nécessairement l'application *HyperAtlas*. De ces envies vont naître deux modules nouveaux que nous présenterons largement dans les chapitres 4 et 5.

3.1.1. Pour une acquisition autonome des données

L'application *HyperAtlas* utilise des jeux de données dans un format binaire spécifique : elle lit des fichiers d'objets sérialisés. Ce format non lisible par un humain est fabriqué à partir d'un ensemble de fichiers textuels, dont la structure et la syntaxe sont formellement établies dans une documentation que connaissent les géographes du projet. L'application *HyperAtlas* ne travaille pas à partir de ces données textuelles directement, parce qu'auparavant, il est nécessaire d'effectuer certaines vérifications et de pré-calculer des informations utiles (sommer des stocks, agréger des contours géométriques) sur les données. Ces informations sont ensuite structurées sous la forme d'un modèle d'objets sérialisés dans des fichiers d'extension « **.hyp** ». Depuis les débuts du projet, l'ingénieur informaticien utilise un programme spécifique pour cette opération d'acquisition de données. Nous expliquons dans le paragraphe suivant pourquoi cette opération ne doit plus être la prérogative de cet acteur du projet.

Le projet touche plusieurs cercles d'utilisateurs, symbolisés dans la figure 17. Nous définissons un cercle d'**utilisateurs** simples, non expert en géomatique ni même savants en matière d'analyse spatiale : ce sont par exemple des politiciens, ou de simples citoyens. Le second cercle est constitué, lui, d'**experts** en géographie, des spécialistes capables d'expliquer l'analyse spatiale aux membres du premier cercle, mais pour autant peu au fait de la technicité sur laquelle elle repose (non aptes à constituer un fond de carte par exemple). Nous y incluons par exemple les points focaux du projet, chargés de relayer notre produit auprès du grand public, et capables de collecter des données statistiques intéressantes des régions géographiques particulières. Et enfin, nous définissons le troisième cercle comme regroupant les experts techniques du projet, ou **administrateurs**, capables de constituer un jeu de données textuelles complet au format adéquat pour chaque module. Ils sont par exemple capables de modifier un maillage, de construire une hiérarchie adaptée pour *HyperAtlas*, etc.

Ces cercles peuvent s'intersecter car les utilisateurs sont souvent polyvalents : par exemple, les experts sont aussi de simples utilisateurs de l'application. L'ingénieur de développement se trouve à l'intersection de ces trois cercles puisqu'il est capable de modifier et de structurer des données géographiques pour l'application, et qu'en même temps, il comprend globalement l'analyse spatiale et que de plus, il joue le rôle d'utilisateur pour vérifier l'application lors de tests unitaires.

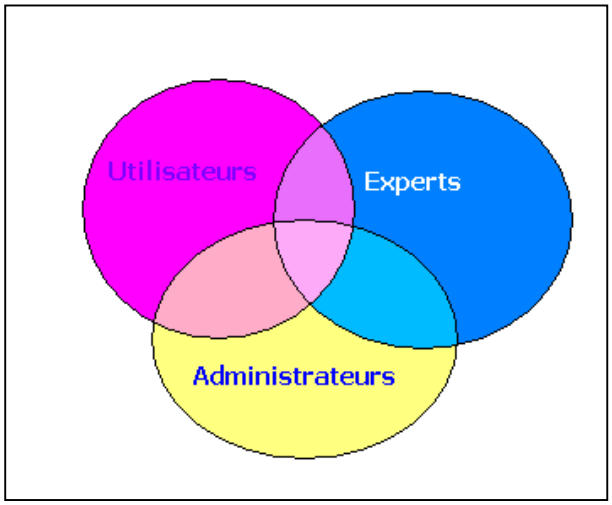


Figure 17. Les cercles d'utilisateurs de HyperAdmin.

Mais il apparaît que les membres de ce troisième cercle sont toujours obligés de s'adresser à l'ingénieur informaticien en charge de *HyperAtlas* pour intégrer leurs données. L'intégration des données consiste à les traduire d'un format textuel en un format binaire d'objets sérialisés, tels que les comprend le logiciel *HyperAtlas*. Ceci constitue un point d'engorgement, et constitue une surcharge de travail pour le développeur qui devrait normalement se consacrer uniquement aux tâches de développement. En effet, il en découle que c'est souvent le développeur qui s'emploie au *packaging* (empaquetage) du logiciel avec les nouvelles données, et à leur mise en ligne pour la diffusion. Il arrivait même parfois que, du fait d'un archivage du projet défaillant et d'un passage de relais mal réalisé entre chaque développeur, des données originales soient perdues, ou encore que nous ne puissions tracer leur provenance.

Il est donc nécessaire de décharger les développeurs de cette tâche fastidieuse, et surtout de donner au troisième cercle d'utilisateurs l'autonomie qu'ils réclament par ailleurs. Les avantages sont immédiats : le taux de diffusion du logiciel augmenterait automatiquement puisque chaque nouveau jeu de données intégré agrandit le cercle des acteurs sociaux ou politiques touchés. D'autre part, les auteurs du jeu de données deviennent alors responsables des données originales et de leur archivage. On peut leur proposer toutefois des outils et un cadre pour cette tâche, afin de centraliser tous les jeux de données. Nous pensons, par exemple, à un système de gestion de contenu (CMS) accessible depuis le Web, ou bien un simple site Web servant de dépôt pour les archives. D'autre part, on peut aussi envisager d'accorder aux utilisateurs du second cercle les moyens d'ajouter ou de modifier des indicateurs statistiques. En effet, relais locaux du produit, ils sont en contact avec des utilisateurs, qui peuvent en retour leur donner accès à des données supplémentaires, motivés par l'intérêt du logiciel.

Le logiciel *HyperAdmin* doit répondre à ce besoin. Il existe un premier prototype mais il ne donne pas satisfaction aux usagers géographes : ceux-ci trouvent son installation complexe et n'arrivent pas à l'utiliser. Dans ces conditions, il paraît pertinent d'abord d'établir un diagnostic et ensuite de déterminer plus exactement les fonctionnalités attendues.

3.1.2. Pour une représentation continue des données

Les membres du projet expriment depuis longtemps leur désir d'avoir un module qui permettrait l'analyse des distributions de phénomènes socio-économiques dans un espace continu. En effet, un grand nombre de phénomènes socio-économiques, et surtout environnementaux, obéissent à des logiques de diffusion et de distribution dans l'espace relevant de fonctions continues, sans extinction brutale le long des frontières.

Par exemple, une cartographie des implantations de type Seveso dans le cadre d'unités administratives de petite taille n'a guère de sens puisque les pollutions éventuelles se propagent de façon continue dans l'espace et ne respectent évidemment pas les limites politiques ou administratives. La cartographie correcte consiste à déterminer la portée spatiale des émanations dangereuses (par exemple une fonction exponentielle négative décroissant de 0 à 100 km) puis, à calculer, en tout point de l'espace, le potentiel d'exposition à une pollution de type Seveso.

La même remarque vaut lorsqu'on étudie le taux de chômage dans un marché du travail non segmenté spatialement (libre circulation des travailleurs) : la meilleure représentation consiste à partir de la distance temps maximale pour se rendre au travail (par exemple, 2 heures de temps routier), puis, à calculer, en tout point de l'espace, le taux de chômage dans un voisinage de 2 heures. L'originalité de ce dernier exemple réside dans le fait qu'un accroissement des mobilités (amélioration des conditions de transport) modifie la cartographie du chômage, ce qui n'est évidemment pas pris en compte dans une cartographie classique par département ou bassin d'emploi.

Pour satisfaire ce besoin, le projet veut mettre en œuvre une méthode originale : le calcul et la visualisation de cartes de potentiel. Cette méthode de généralisation de l'information s'utilise pour projeter un espace maillé connu sur un espace géométrique de représentation. Les données traitées proviennent de comptages sur des unités territoriales du maillage initial. Il faut retenir qu'avec cette méthode, nous pourrons nous affranchir des maillages spatiaux et dégager la structure spatiale des phénomènes sociaux, économiques, environnementaux, ou même épidémiologiques, et ceci indépendamment du niveau de récolte des données. Cette méthode d'analyse peut-être qualifiée de **multiscalaire** dans le sens où les résultats des analyses peuvent être comparés à plusieurs échelles lorsque l'on fait varier le paramètre de portée qui, grossièrement, peut se comparer à un rayon d'analyse. Nous définirons ultérieurement ce paramètre lorsque nous expliquerons les fondements mathématiques de la méthode.

Cette plate-forme sera destinée à différents acteurs de l'analyse territoriale (chercheurs en géographie, en sciences sociales et humaines, mais aussi décideurs politiques et grand public) et acteurs de l'aménagement du territoire. Il s'agit notamment de répondre à une demande applicative aiguë exprimée par divers organismes de production de données socio-économiques tels que la DIACT au niveau national, ou EUROSTAT pour l'Europe. Les applications visées se situent principalement dans le domaine socio-économique ou environnemental et l'aide à la décision en matière de prospective territoriale, qui sont les thèmes principaux du projet Hypercarte.

Le module *HyperSmooth* est une réalisation qui doit répondre à ce besoin. Nous espérons pouvoir générer « à la volée » des cartes de potentiel, en utilisant un client interactif permettant le paramétrage les analyses et la génération d'un atlas de cartes de façon aussi agréable que *HyperAtlas*.

3.2. HyperAtlas : un module pour l'analyse territoriale multiscalaire

Nous décrivons ici les fonctionnalités offertes par l'application *HyperAtlas*, et surtout l'architecture sur laquelle elle repose. En effet, nous proposons de dériver cette architecture dans les prochains modules qui doivent satisfaire les besoins exposés précédemment.

3.2.1. Fonctionnement de l'interface

Nous avons vu en introduction qu'*HyperAtlas* permet une analyse territoriale multiscalaire sur des indicateurs sociaux-économiques, ou autres, sur différents espaces d'études. Nous précisons ici les modalités de cette analyse.

Avant toute visualisation, l'utilisateur doit spécifier :

- L'*espace d'étude* : il s'agit du territoire spécifié par le fichier d'entrée dans son intégralité ou d'une portion de celui-ci.
- Le *niveau hiérarchique* du maillage qui définit dans le maillage le niveau des unités territoriales qui seront affichées sur les cartes proposées.
- Le *ratio* sur lequel porte l'analyse spatiale proposé par HyperAtlas. Ce ratio est formé par deux stocks ou indicateurs : l'un constitue le numérateur, l'autre le dénominateur.

Cette sélection s'opère dans la partie supérieure de l'interface (figure 18).

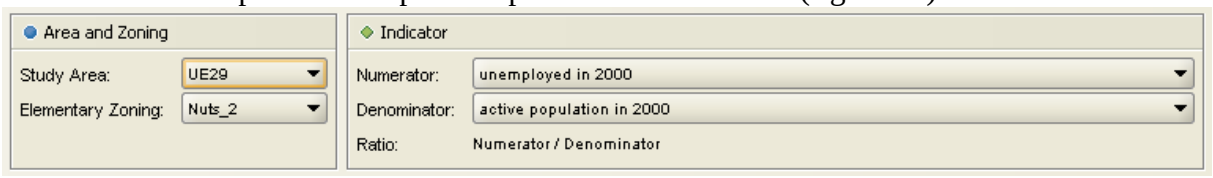


Figure 18. Paramétrage de l'aire d'étude, du maillage et du ratio.

La partie inférieure et centrale est consacrée à l'affichage des cartes. *HyperAtlas* affiche huit cartes, chacune accessible depuis un onglet. Pour chaque carte, des explications et des options de personnalisation (choix de couleurs, nombre de classes, etc.) sont fournies. Au sein de chaque carte, zoom et déplacement sont possibles.

Description du contenu des cartes :

- La carte *Area and Zoning* affiche l'espace d'étude et les unités territoriales dont il est composé au niveau hiérarchique sélectionné (figure 19).

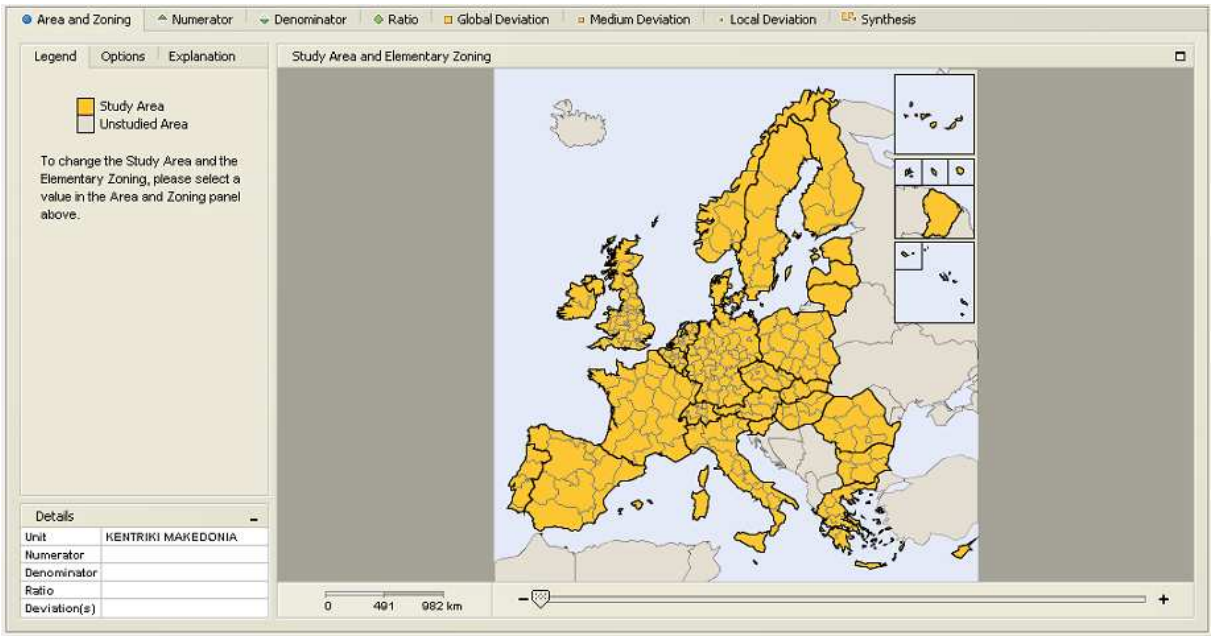


Figure 19. Première carte de l'ensemble des onglets : elle présente le contexte d'étude.

- La carte *Numerator* affiche, pour chaque unité territoriale du niveau hiérarchique sélectionné, un disque de taille proportionnelle à la valeur du stock constituant le numérateur du ratio. La couleur du disque peut être choisie, cf. figure 20.
- La carte *Denominator* affiche, pour chaque unité territoriale du niveau hiérarchique sélectionné, un disque de taille proportionnelle à la valeur du stock constituant le dénominateur du ratio. La couleur du disque peut être choisie, cf. figure 20.
- La carte *Ratio* affiche, pour chaque unité territoriale du niveau hiérarchique sélectionné, la couleur correspondant à la classe d'appartenance du ratio. Une classe est définie par un intervalle de valeurs du ratio. Le nombre de classes et la couleur du dégradé utilisé pour différencier les classes peuvent être fixés, cf figure 21.

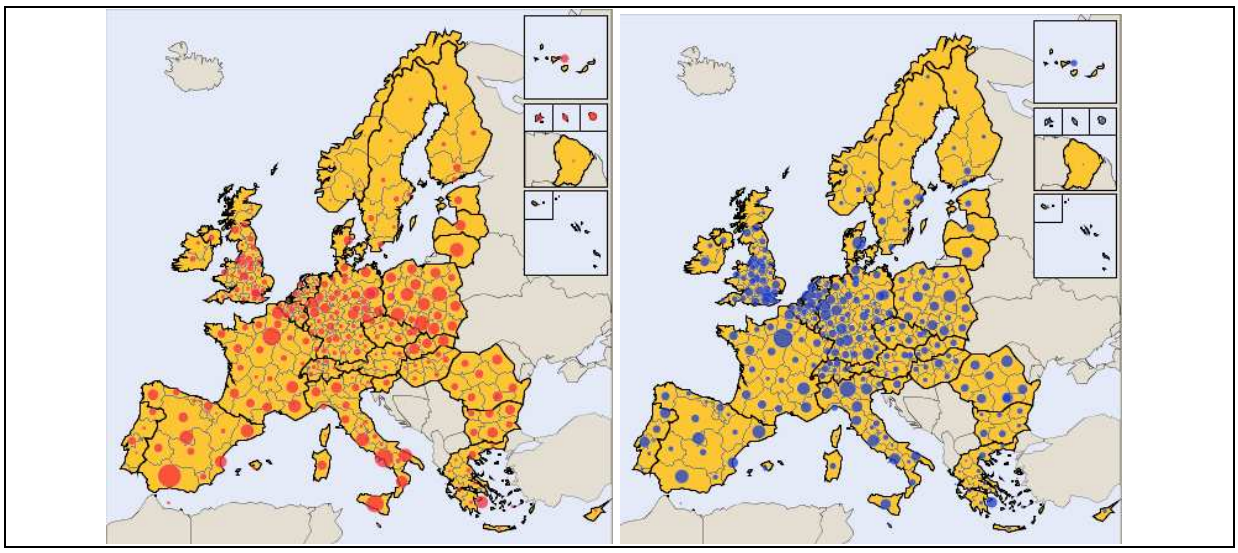


Figure 20. Cartes à cercles proportionnels pour le numérateur et le dénominateur

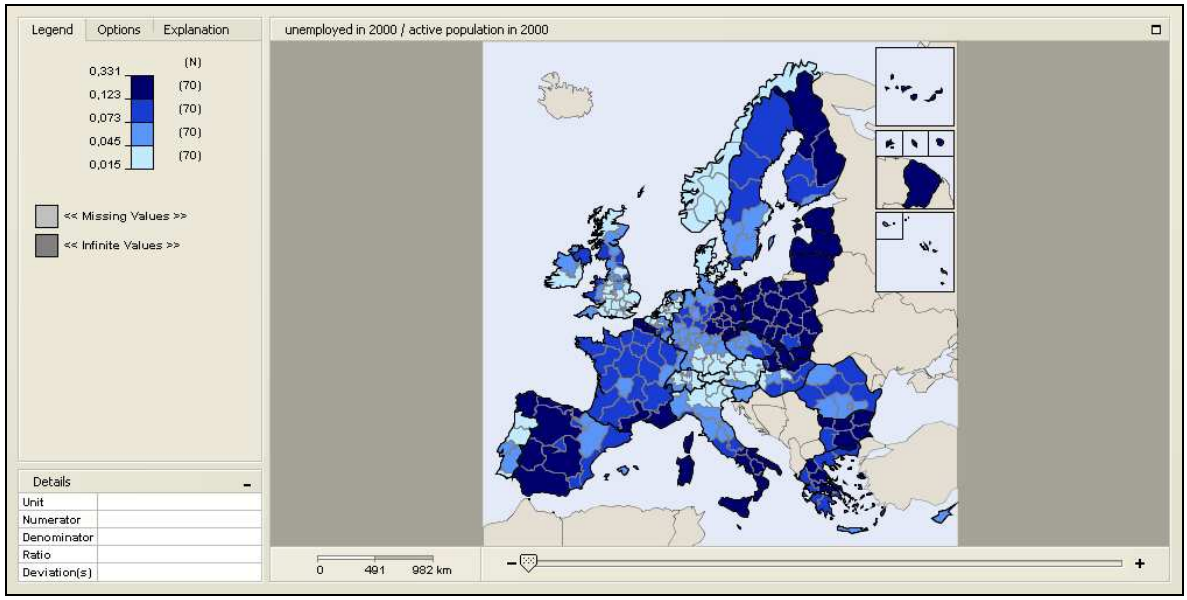


Figure 21. Carte choroplèthe présentant le ratio.

Les trois autres cartes affichent des déviations. Une déviation est un rapport exprimé en pourcentage d'un ratio à une moyenne calculée en fonction du contexte de référence. Ce contexte peut être de trois types : global, intermédiaire ou local, et il est choisi dans le panneau « *Context for the Deviations* », montré par la figure 22.

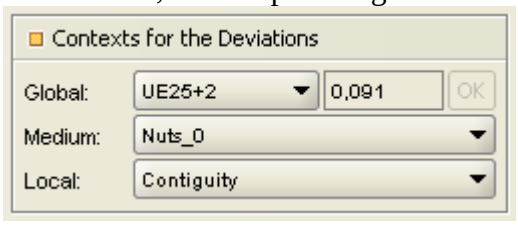


Figure 22. Panneau de sélection du contexte des déviations.

- La carte *Global Deviation* affiche, pour chaque unité territoriale du niveau hiérarchique sélectionné, la couleur correspondant à la classe d'appartenance du pourcentage de déviation du ratio de l'unité territoriale par rapport à une moyenne qui dépend du contexte global. Ce contexte est soit un espace de référence dans son intégralité et choisi parmi ceux disponibles, soit une valeur à fixer, cf. figure 23.
- La carte *Medium Deviation* affiche, pour chaque unité territoriale du niveau hiérarchique sélectionné, la couleur correspondant à la classe d'appartenance du pourcentage de déviation du ratio de l'unité territoriale par rapport à une moyenne évaluée dans un contexte intermédiaire. Ce contexte intermédiaire est constitué d'une unité territoriale supérieure englobante, dont le niveau hiérarchique dépend du choix de l'utilisateur, cf. figure 23.
- La carte *Local Deviation* affiche, pour chaque unité territoriale du niveau hiérarchique sélectionné, la couleur correspondant à la classe d'appartenance du pourcentage de déviation du ratio de l'unité territoriale par rapport une moyenne évaluée dans un contexte local. Ce contexte local est constitué par l'ensemble des unités territoriales voisines (mitoyennes) et de même niveau hiérarchique, cf. figure 23

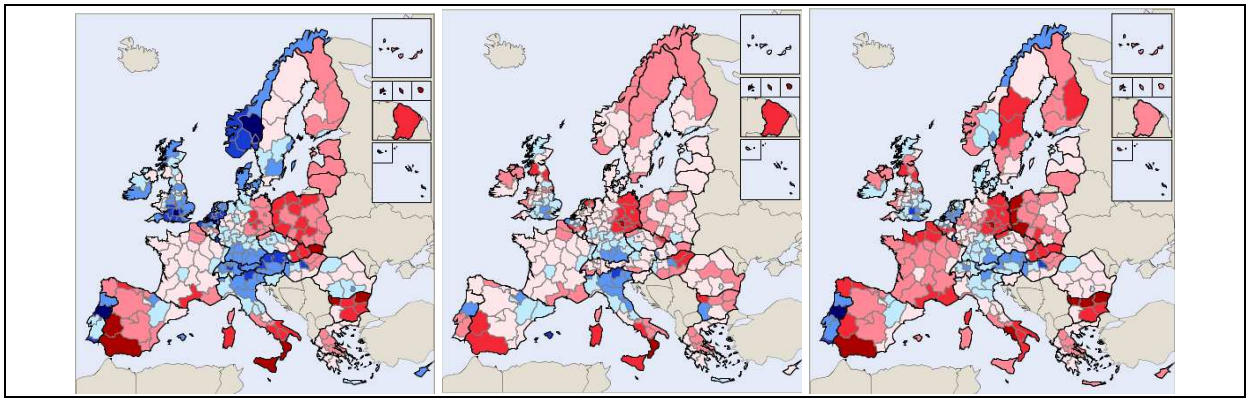


Figure 23. Cartes choroplèthes pour la déviation globale, la déviation moyenne et la déviation locale.

Enfin une carte de synthèse est disponible, (cf. figure 24), qui affiche, pour chaque unité territoriale du niveau hiérarchique sélectionné, la couleur correspondant à la classe d'appartenance de la synthèse des trois déviations (globale, intermédiaire et locale) du ratio. A partir de cette carte, il est possible d'obtenir un diagramme en bâtons montrant les trois déviations du ratio en sélectionnant sur une unité territoriale.

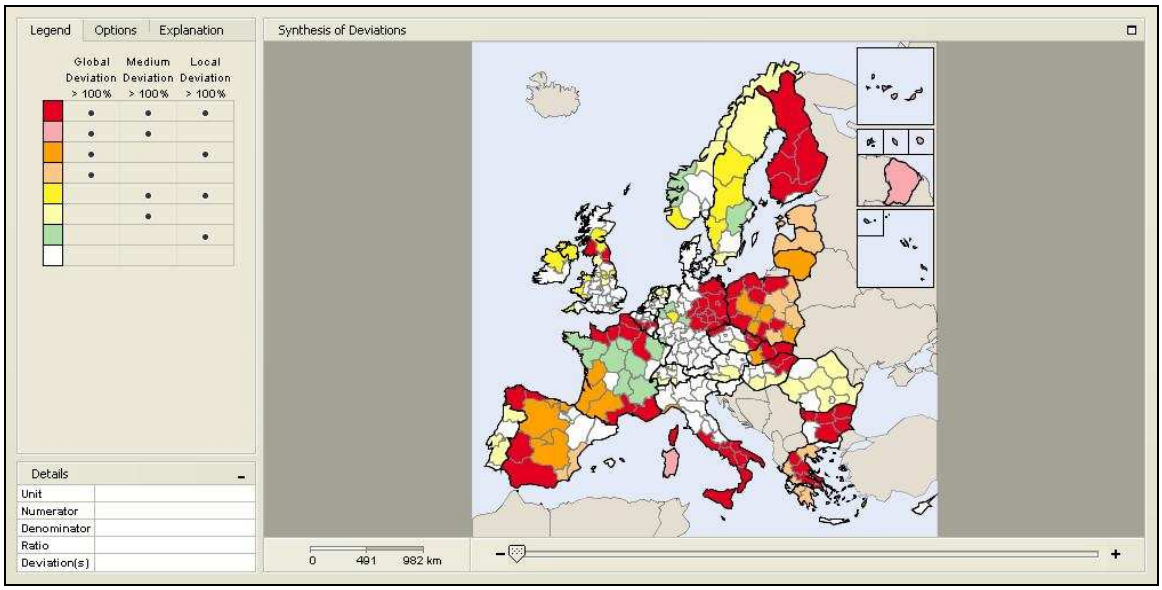


Figure 24. Carte choroplèthe de synthèse.

L'application *HyperAtlas* permet à l'utilisateur de sauvegarder les paramètres de l'analyse effectuée et de produire un rapport sous format HTML incluant les paramètres de l'analyse (aire d'étude, maillage considéré, contexte des déviations), et le résultat de l'analyse : un tableau des indicateurs calculés par unité territoriale, ainsi que les images au format PNG ou JPEG des huit cartes produites. Mais elle ne propose pas encore de moyens simples pour importer ses propres données, comme GéoClip ou bien PhilCarto. D'autre part, toutes les cartes sont basées sur des maillages, et aucune forme de représentation continue (par carroyage, isolignes, etc.) n'est disponible, contrairement au Cartographeur.

3.2.2. Architecture du logiciel

HyperAtlas, développé en langage Java, présente une structure en couches : une couche graphique basée sur l'utilisation de la bibliothèque SWING (la vue), une couche métier comprenant les données et leur manipulation pour effectuer les divers calculs de l'analyse territoriale multi-scalaire (le modèle), et une couche technique contrôlant les échanges entre la vue et le modèle via un système d'évènements logiciels (le contrôleur). Cette structure ressemble à celle du *Model View Controler* (MVC) que l'on retrouve dans certains schémas de développement proposés pour des applications Web.

Nous présentons ici très succinctement les trois couches qui ont déjà été largement documentées dans les précédents mémoires sur le projet HyperCarte, [Martin, 2004], [Cuenot, 2005], et [Chabert, 2007]. Ces rappels doivent aider à comprendre les extensions de code apportées dans les modules *HyperAdmin* et *HyperSmooth*.

a) Les composants graphiques dans HyperAtlas

La couche graphique repose sur Java2D : le paquetage *java.swing* est utilisé pour écrire les composants graphiques, et la bibliothèque graphique *JGoodies* sert à améliorer le rendu graphique de l'application. Pour obtenir le rendu de la figure 2, les divers composants s'imbriquent dans une composition que présente la figure 25.

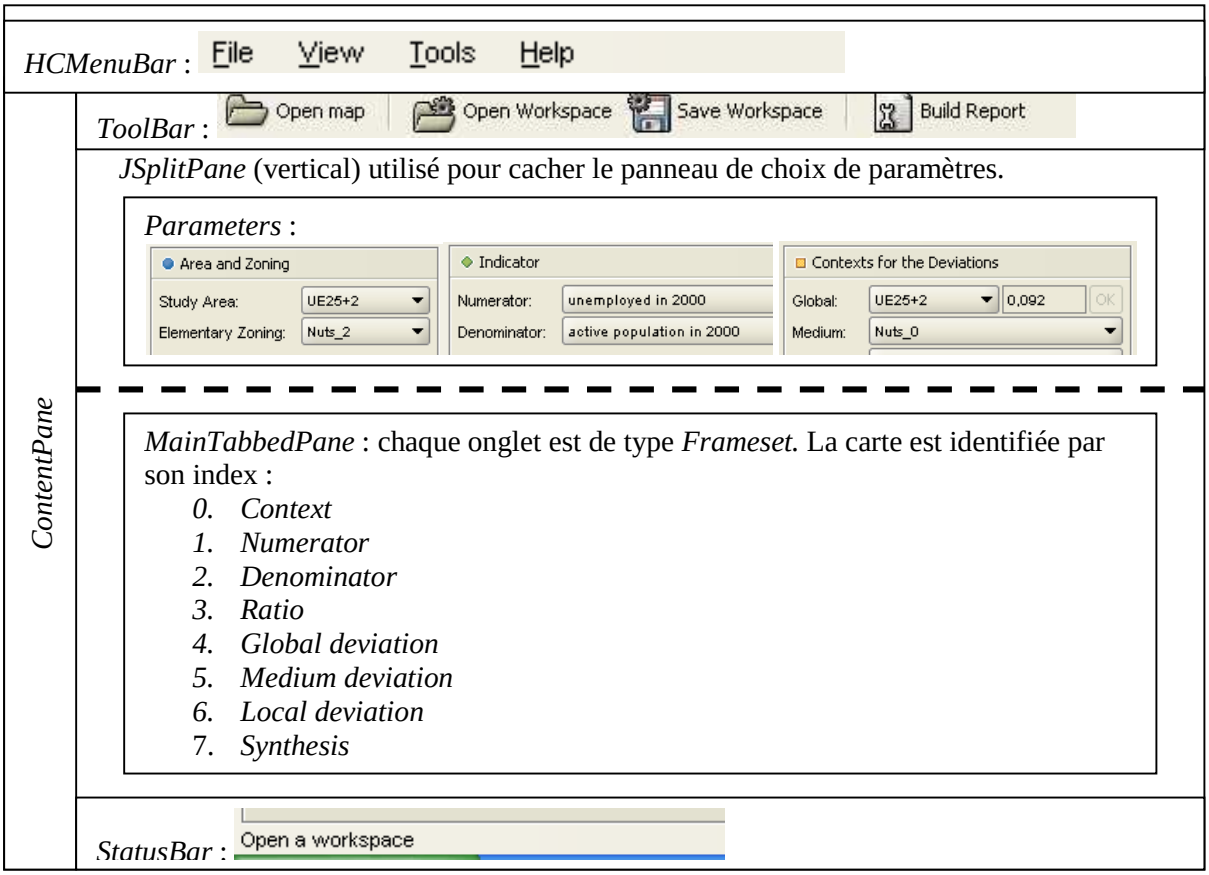


Figure 25. Composition des éléments graphiques de l'interface.

Le composant *Parameters* est responsable du paramétrage de l'analyse : choix du contexte (aire d'étude, niveau de maillage), choix des indicateurs (numérateur et dénominateur) et choix des contextes pour les trois types de déviation. Ces choix se font à partir de listes déroulantes, excepté pour la déviation globale qui peut aussi être spécifiée dans un champ texte.

L'objet graphique *MainTabbedPane* hérite de *JTabbedPane* et permet de gérer l'ensemble des cartes sous formes d'onglets sélectionnables. Il utilise à cette fin un objet *Frameset* qui contient autant d'onglets (*tab*) que de cartes : 8 en tout. Chaque carte est accessible par un index variant de 0 à 7.
Lorsqu'un utilisateur sauve une carte dans *HyperAtlas*, c'est l'image peinte par l'objet graphique *Frameset* qui est exportée. Cette image peut-être exportée au format JPEG ou PNG selon le choix de l'utilisateur. Les figures 19, 21 et 24 en sont des exemples. La figure 26 détaille la constitution de chaque onglet.

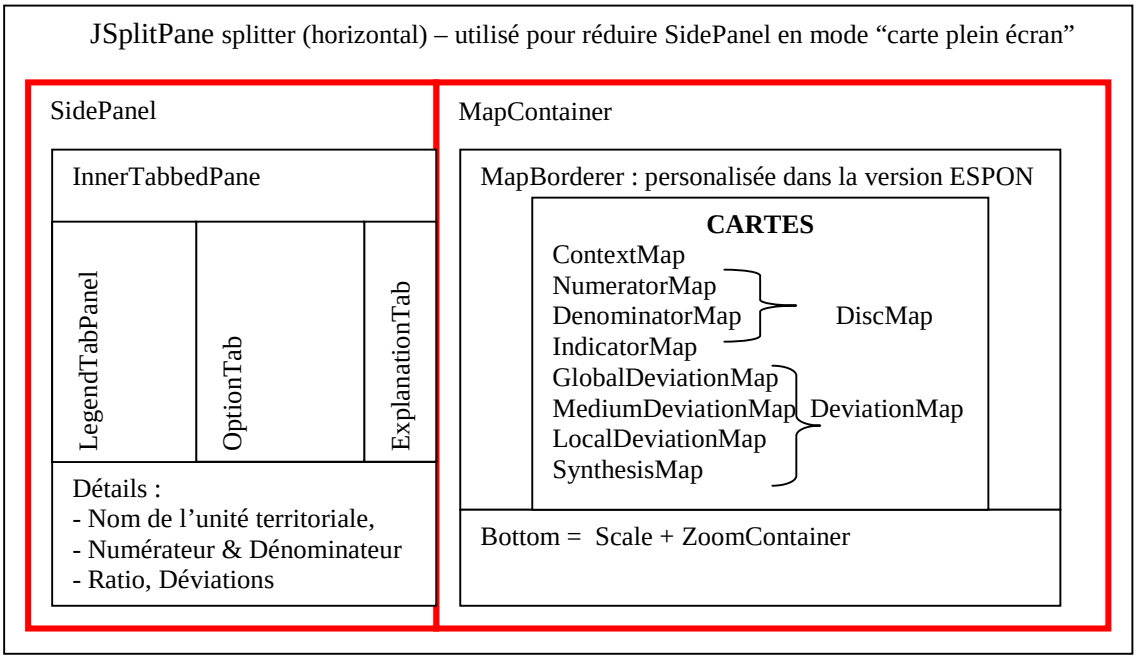


Figure 26. Composition de *Frameset.*

InnerTabbedPane est aussi un composant graphique qui hérite de *JTabbedPane* : c'est un support pour des onglets graphiques, accessibles par leur index. Il instancie trois onglets pour chaque carte:

- la légende : *LegendTabPanel*, qui contient un objet *JPanel* de type Legend
- les options : *OptionsTab* qui contient un objet *JPanel* de type Options
- les explications : *ExplanationTab* qui hérite de l'objet *JPanel*

Ces onglets sont adaptés en fonction de la carte que l'on représente : carte à cercles proportionnels, carte choroplèthe. Les cartes du numérateur et du dénominateur sont des cartes à cercles proportionnels, tandis que les autres cartes (aire d'étude, ratio, les trois déviations, la synthèse) sont des cartes choroplèthes. Le type du composant graphique spécialisant *Legend* ou *Options* dépend de l'index de la carte affichée. Pour l'onglet

ExplanationTab, la valeur du texte affiché est adaptée en fonction de l'index passé en paramètre. Le tableau 3 synthétise ces informations.

Map index	Cartes	Type de Map	Legend	Options
0	ContextMap	Map	ContextLegend	ContextPalettOptions
1	NumeratorMap	DiscMap	DiscLegend	DiscOptions
2	DenominatorMap			
3	IndicatorMap	Map	SimplePalettsLegend	SingleHueOptions
4	GlobalDeviationMap	DeviationMap		DoubleHueOptions
5	MediumDeviationMap			
6	LocalDeviationMap			
7	SynthesisMap		SynthesisLegend	SynthesisOptions

Tableau 3. Liste des cartes, et types d'option et de légende associés.

b) La couche métier de HyperAtlas

L'application exploite un modèle de données objet qu'elle lit dans des fichiers sérialisés. Ce modèle est conçu pour accélérer les calculs permettant de mener une analyse, calculs qui sont tous reportés dans une couche métier de l'application et qui permettent pour chaque unité territoriale :

- le calcul d'un ratio,
- le calcul de la déviation globale, moyenne et locale,
- le calcul d'une synthèse.

Ces calculs concernent aussi pour l'ensemble des unités, sur une aire d'étude et un maillage donné :

- l'évaluation de la somme totale d'un indicateur,
- la détermination de la valeur minimale, maximale de l'indicateur,
- la discrétisation du ratio, et des déviations pour construire les cartes choroplèthes.

Les formules permettant de calculer les déviations globales, moyennes et locales sont documentées dans le mémoire de Philippe Martin, [Martin, 2004]. Tous ces calculs s'effectuent dans la classe *hypercarte.hyperatlas.Logic*.

Nous décrivons ici le modèle de données de façon générique, puis nous détaillons la structure du paquetage *hypercarte.data* qui était employé lors du début de ce mémoire. Les évolutions du modèle, ainsi que la structure du nouveau modèle de données mis en place (*hypercarte.hyperatlas.serials*) sont documentées dans le chapitre suivant.

L'objet central du modèle de données est **l'unité territoriale** (*unit*): en effet, les analyses de HyperAtlas traitent des partitions d'espace géographique fabriquées à partir de **maillages territoriaux** (*zonings*), généralement administratifs. Les mailles qui composent ces partitions sont désignées par le terme unité territoriale, et les différents maillages forment une hiérarchie, où les maillages à de plus haut niveau sont composés par l'agrégation des mailles de niveau immédiatement inférieur, et ainsi de suite, (cf. figure 1). Les mailles de même niveau entretiennent entre elles des relations de voisinage, que pour l'instant nous limitons à

la contiguïté. Ensuite, HyperAtlas manipule des **aires d'étude** (*area*s) qui sont la définition d'un sous-ensemble de mailles de même niveau. Par exemple, l'aire « PECO » (Pays de l'Europe Centrale et Orientale) regroupe les mailles de niveau NUTS 0 (pays) appartenant grosso modo aux pays de l'Est avant la chute du mur de Berlin en 1989. L'aire « Arc Atlantique » regroupe les mailles de niveau NUTS 3 bordant la côte Atlantique de l'Europe, entre l'Irlande et le Portugal. Enfin, HyperAtlas propose d'analyser des indicateurs (ou **stocks**), statistiques résultantes de comptage sur les mailles. Ces stocks ont une propriété intéressante : ils sont additifs, c'est-à-dire que la valeur d'un stock pour une maille de rang *n* est égale à la somme de ce stock sur les mailles de rang *n-1* qui la composent. L'ensemble de ces données est regroupé dans un **projet** (*project*). La figure 27 donne une vue générale de cette composition et des relations entre objets.

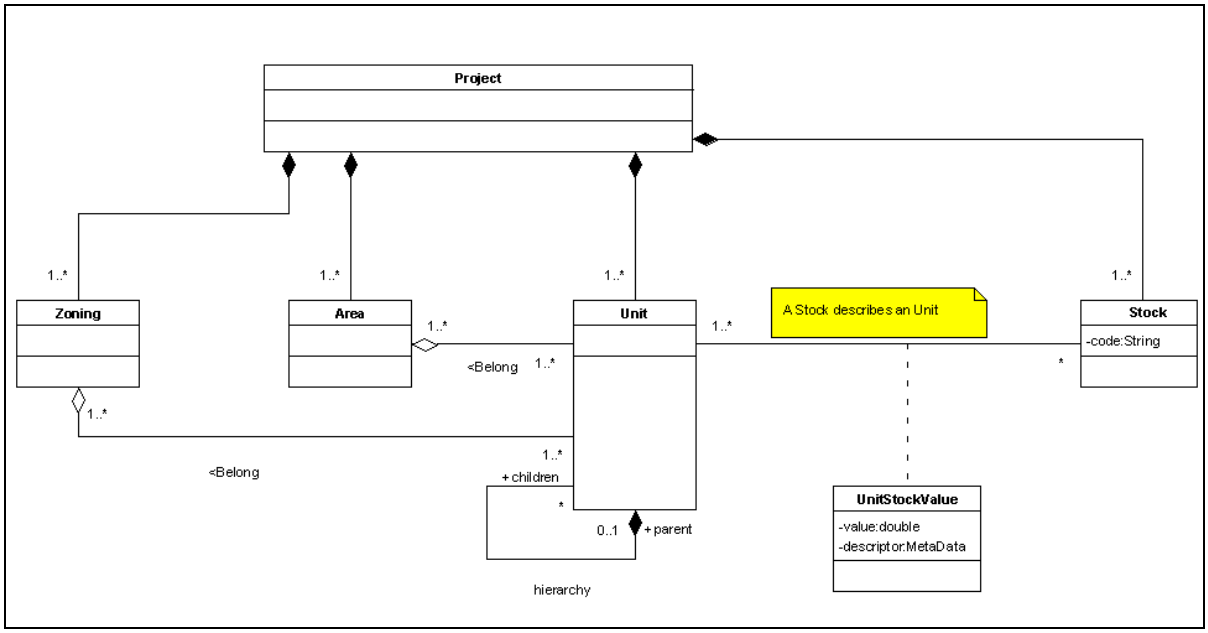


Figure 27. Vue générale des relations entre objets du modèle de données.

La propriété d'additivité des stocks, ainsi que le fait que nous utilisions une hiérarchie de maillages strictement emboîtés nous permet de développer le jeu de données sur tous les niveaux hiérarchiques à partir seulement du niveau le plus inférieur : les stocks s'additionnent, les contours géométriques s'agrègent. Cependant, les relations de distance se recalculent sur chaque niveau.

Le paquetage *hypercarte.data* reprend exactement le modèle de données de la figure 27 et de plus donne un identifiant de type chaîne de caractères à toutes les entités du modèle, qui est le code attribué dans les données textuelles originelles. Certaines informations (comme les relations de contiguïté ou bien la valeur des stocks pour les mailles de niveau supérieur) sont toujours recalculées lorsque la classe *Logic* les lit.

c) La couche technique contrôlant HyperAtlas

Il s'agit de décrire comment interagissent le modèle métier et la couche graphique. Ces interactions ne sont pas centralisées par un contrôleur unique comme dans Struts, où une

servlet *ActionControler* contrôle les échanges entre la vue et le modèle à l'aide de règles d'interaction qui sont écrites par le développeur dans un fichier de configuration unique. [Cavaness, 2002]

En fait, les échanges reposent sur un système d'émission et de réception d'évènements hérité des réalisations de Thierry Bissler [Bissler, 2003] : les actions et les choix de l'utilisateur sur l'interface graphique provoquent la réaction des composants graphiques qui jouent alors le rôle d'émetteur d'évènements logiciels spécifiques auxquels pourront réagir tous les abonnés simultanément. Ils s'adressent à un objet unique, central dans ce système, le répartiteur d'évènement, *hypercarte.hyperatlas.event.Dispatcher*, qui notifie les abonnés via une méthode spécifique (*fireEvent(…))* prenant en paramètre l'évènement produit. Les évènements logiciels traités par le répartiteur sont définis dans des spécialisations de la classe *AbstractEvent.* Plus tard, lorsque nous voudrons faire gérer des évènements spécifiques aux modules HyperAdmin ou HyperSmooth, il suffira de spécialiser cette classe.

Toutefois la mise à jour de l'interface graphique et les actions sur le modèle de données nécessitent en général la connaissance de plusieurs paramètres, et il existe un point d'accès centralisé aux paramètres généraux de l'application (comme le niveau de maillage, l'aire d'étude en cours, le niveau de zoom, l'échelle, etc.), c'est l'objet *hypercarte.hyperatlas.config.Settings*. Cette classe, instanciée une seule fois de façon statique, contient une liste de champs constituant le paramétrage global de l'application, champs privés auxquels on accède par des accesseurs, en lecture (*getX()*) ou en écriture (*setX(…)*). Les composants métiers et graphiques utilisent *Settings* pour enregistrer ou consulter le paramétrage. Ultérieurement, pour stocker le paramétrage spécifique à *HyperAdmin* ou bien *HyperSmooth*, nous exploiterons le même système mais dans des classes dédiées afin de ne pas alourdir *Settings* avec des informations inutiles pour *HyperAtlas* :

- *hypercarte.hyperadmin.config.HASettings*,
- *hypercarte.hypersmooth.config.HSSettings*.

En annexe, dans la partie consacrée à l'ingénierie logicielle, nous décrivons aussi comment sont gérés les codes sources du logiciel, leur organisation, et les outils de gestion de configuration mis en place. On y trouve aussi une section concernant les développements communs aux trois modules, portant sur l'internationalisation et la journalisation des applications.

4. HyperAdmin : un module d'acquisition de données

Dans le précédent chapitre, nous avons expliqué qu'il était nécessaire d'offrir aux acteurs du projet la possibilité d'intégrer eux-mêmes des jeux de données pour *HyperAtlas*. Il existe un module prototype pour l'acquisition des données, nommé *HyperAdmin*, qui ne donne pas entière satisfaction. L'objet de ce chapitre est de présenter notre contribution à ce module. Dans un premier temps, nous décrivons les besoins que ce module doit satisfaire, et nous établissons une révision du cahier des charges qui tient compte des demandes d'évolution émises par les membres du projet. Nous analysons alors le module existant pour déterminer ses failles. Dans un deuxième temps, nous exposons notre proposition technique destinée à combler les failles observées, tout en profitant au maximum du code existant. Dans un troisième temps, nous présentons notre réalisation en détail, et le fonctionnement du nouveau module. Enfin, nous concluons ce chapitre par un bilan de notre action sur *HyperAdmin*, en rapport avec les objectifs visés.

4.1. Définition du cahier des charges

HyperAdmin a pour objectif principal l'acquisition, le stockage et la gestion des données utilisables par le logiciel *HyperAtlas*. Nous détaillons dans ce paragraphe les besoins que le logiciel devrait couvrir à terme, et critiquons le prototype existant de façon à déterminer les axes prioritaires d'amélioration en relation avec notre cahier des charges.

4.1.1. Révision du cahier des charges

Notre contribution a débuté par une phase de remise en question des spécifications du logiciel. Ce questionnement, doublé d'un travail d'imagination, nous a amené à réviser le cahier des charges existant, et donner un nouvel ordre de priorité aux fonctionnalités.

a) Statut du prototype par rapport à l'ancien cahier des charges

Nous listons ici les différentes attentes qui avaient été émises pour ce prototype, *HyperAdmin V0.1 Alpha 2*, et nous vérifions si elles sont satisfaites.

Premièrement, *HyperAdmin* a pour rôle principal de **générer des fichiers de données** dans un format spécifique pour *HyperAtlas*. Dans le précédent chapitre, nous avons décrit de façon générique le modèle de données sur lequel s'appuie le logiciel, ainsi que le paquetage de données sérialisées correspondant, *hypercarte.data*. Le prototype remplit très bien ce rôle : il permet d'acquérir des données pour *HyperAtlas* à partir de fichiers textes ou Excel, et lit des fonds de cartes fournis au format MIF/MID. Ces données, groupées sous une entité nommée « projet », sont enregistrées dans une base de données. Ce projet peut-être ré-ouvert ultérieurement pour lui apporter des modifications, et peut s'exporter dans un fichier de données sérialisées d'extension « .hyp » pour *HyperAtlas*.

Deuxièmement, il était demandé de pouvoir **visualiser directement les données** dans une interface graphique, afin que l'utilisateur puisse contrôler le fond de carte importé, vérifier les maillages, visualiser les aires d'études et les valeurs des stocks sur les unités. Ce besoin est aussi satisfait, et va même au-delà de la demande car le prototype reprend toutes les

fonctionnalités de HyperAtlas en permettant de réaliser une analyse territoriale multi-scalaire spatiale. Il utilise en effet la même interface graphique que HyperAtlas, et montre les cartes de ratio et déviation. Nous notons que ces cartes ne sont pas utiles pour vérifier l'exactitude ou la complétude des données lues.

Troisièmement, l'application devait aussi permettre **d'optimiser les calculs** de HyperAtlas en préparant dans les jeux de données certaines informations utiles :

- calculer d'avance les contours géométriques d'une unité territoriale de niveau supérieur par agrégation des contours des unités la composant,
- pré-déterminer les relations de contiguïté entre les unités.

Ces optimisations sont mises en œuvre dans *HyperAdmin*, à la lecture des données depuis les fichiers texte, ou bien à l'ouverture d'un projet en base de données, mais ne sont cependant pas reportées dans le modèle de données sérialisées, du paquetage *hypercarte.data*.

Quatrièmement, l'application HyperAdmin doit permettre d'enrichir des jeux de données existant afin d'**ajouter ou de modifier des données**. La demande portait initialement et principalement sur les stocks, dont parfois les valeurs manquent, ou bien sont révisées par les instituts de statistiques. Cette fonctionnalité n'est pas disponible, mais la conception du module prévoit son implantation.

Cinquièmement, HyperAdmin devait donner les moyens à l'administrateur de générer des jeux de données restreints, résultant d'un **filtrage** sur les projets importés initialement. Ce filtrage pouvait permettre par exemple de limiter les indicateurs à certaines années d'extraction, ou bien de limiter le nombre de maillages ou d'aires d'étude disponibles dans le fichier produit pour *HyperAtlas*. Cette fonctionnalité n'est pas non plus présente, cependant la structure de *HyperAdmin* anticipe ce besoin.

Sixièmement, afin d'autoriser une **gestion thématique des indicateurs**, il faudrait qu'*HyperAdmin* supporte la gestion de méta-données (des données descriptives renseignant la provenance, le thème, la date d'extraction, etc.). Ces méta-données pourraient permettre d'effectuer des extractions thématiques reposant sur l'utilisation d'ontologies. Cette perspective est abordée dans le cahier des charges de Christophe Chabert [Chabert, 2007], et le modèle de données relationnel définit des thèmes ainsi que des relations thème-stock, cependant aucun mécanisme d'exploitation pour ces données descriptives n'est implémenté dans le module *HyperAtlas*.

Septièmement, HyperAdmin devait gérer des comptes utilisateurs pour la mise en place de **droits d'accès aux données**. Ces droits concernent la lecture, modification, création, ou suppression de projet. L'ambition originale était de gérer les droits utilisateurs de la même façon que dans le système Unix, avec le regroupement des utilisateurs dans des groupes ayant des profils de droits identiques, chaque individu héritant des droits de son groupe, ou bien spécialisant un droit. Les prémisses d'un mécanisme de gestion de protection des données existent (une relation associant un projet à un utilisateur créateur du projet, et cet utilisateur est lié à une organisation). Cependant, ces deux relations sont insuffisantes pour mettre en place un vrai contrôle d'accès sur les données.

Et dernièrement, le projet *hypercarte* porte une attention particulière à l'ergonomie de l'interface et son ambition est de la rendre **adaptable à divers profils d'utilisateurs**, sachant

qu'*HyperAtlas* s'adresse à un public aux compétences très variées. A cette fin, il faudrait que le module *HyperAtlas* puisse prendre en compte leurs préférences, leurs niveau de connaissances, la configuration matérielle utilisée, et ces profils devrait être enregistrés par *HyperAdmin*. Ce dernier point n'est pas du tout traité par le module HyperAdmin, et peut-être nécessite-t-il comme pré requis une étude préalable sur l'exploitation des profils utilisateurs et la mise en œuvre de l'adaptabilité dans *HyperAtlas.*

b) Demandes d'évolution d'HyperAdmin

Suite à la présentation du premier prototype *HyperAdmin V0.1 Alpha 2* de Christophe Chabert, en février 2006, les membres du projet nous font part de certaines idées nouvelles concernant le logiciel.

Au préalable, avant d'examiner ces dernières requêtes, certaines mentionnées précédemment sont révisées. Par exemple, le point concernant le contrôle d'accès aux données ne sera plus implémenté dans *HyperAdmin* : nous nous reposerons sur la gestion de droits d'accès implémentée dans la base de données PostgreSQL. L'utilisation d'un accès protégé par mot de passe au serveur sera suffisante pour l'instant. Nous avons aussi préféré ignorer les attentes concernant l'adaptabilité de l'application, avec la gestion de profils utilisateurs, puisque d'une part elles n'avaient pas été renouvelées à notre arrivée, et que d'autre part, il ne nous semblait pas possible de concevoir une architecture adaptée dans les délais, au vu de la difficulté estimée.

D'abord, il faudrait que le logiciel accorde plus de liberté au concepteur du jeu de données. En effet, celui-ci est actuellement très guidé par l'application qui lui demande de générer le jeu de données en une seule fois : fond de cartes, hiérarchies et indicateurs doivent tous être fournis en même temps. Le format des fichiers de données est documenté dans un manuel que l'on a déposé sur InriaGForge, et qui est diffusé auprès des experts du projet. Le contenu de ce manuel est aussi repris dans le mémoire de Christophe Chabert, [Chabert, 2007] et c'est pourquoi nous ne nous attardons pas dessus.
Le logiciel guide l'utilisateur afin qu'il fournisse en trois étapes :

- une description générale du projet (nom, identifiant, échelle du fond de carte, langue de description des entités) que montre la figure 28;
- le type des fichiers ASCII : texte ou Excel. Un type mixte (avoir les indicateurs dans un fichier Excel, et la structure géographique dans un ensemble de fichiers texte) n'est pas permis, (cf. figure 29) ;
- le chemin vers le répertoire contenant les fichiers, de façon globale, ou bien en précisant les répertoires pour chaque fichier (cela présuppose que les fichiers Excel sont au nombre de 2, un pour le fichier décrivant les indicateurs, un autre pour la structure de données), (cf. la figure 30).

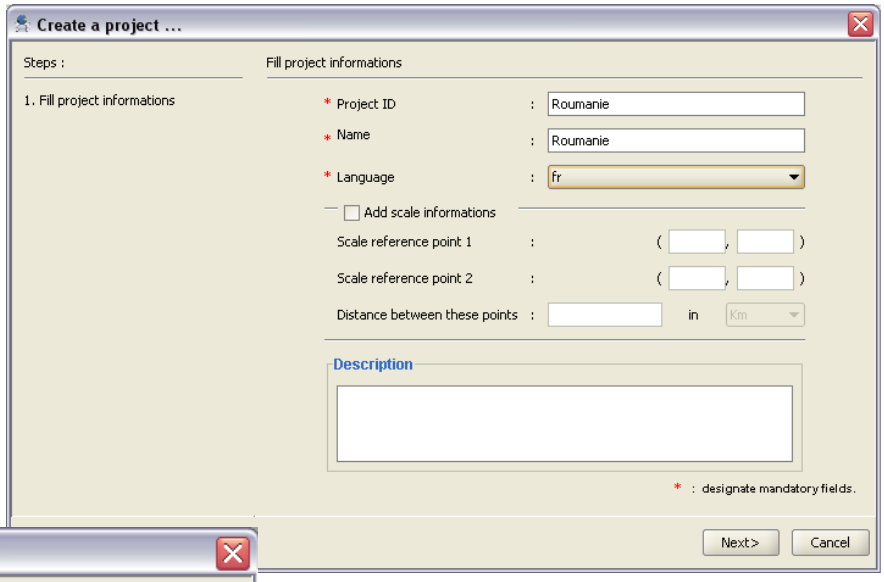


Figure 28. Première étape pour l'import de données avec *HyperAdmin V0.1 Alpha 2.*

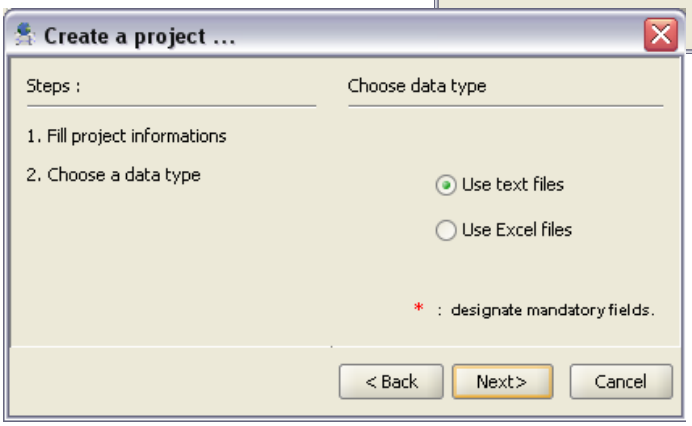


Figure 29. Deuxième étape pour l'import de données avec *HyperAdmin V0.1 Alpha 2.*

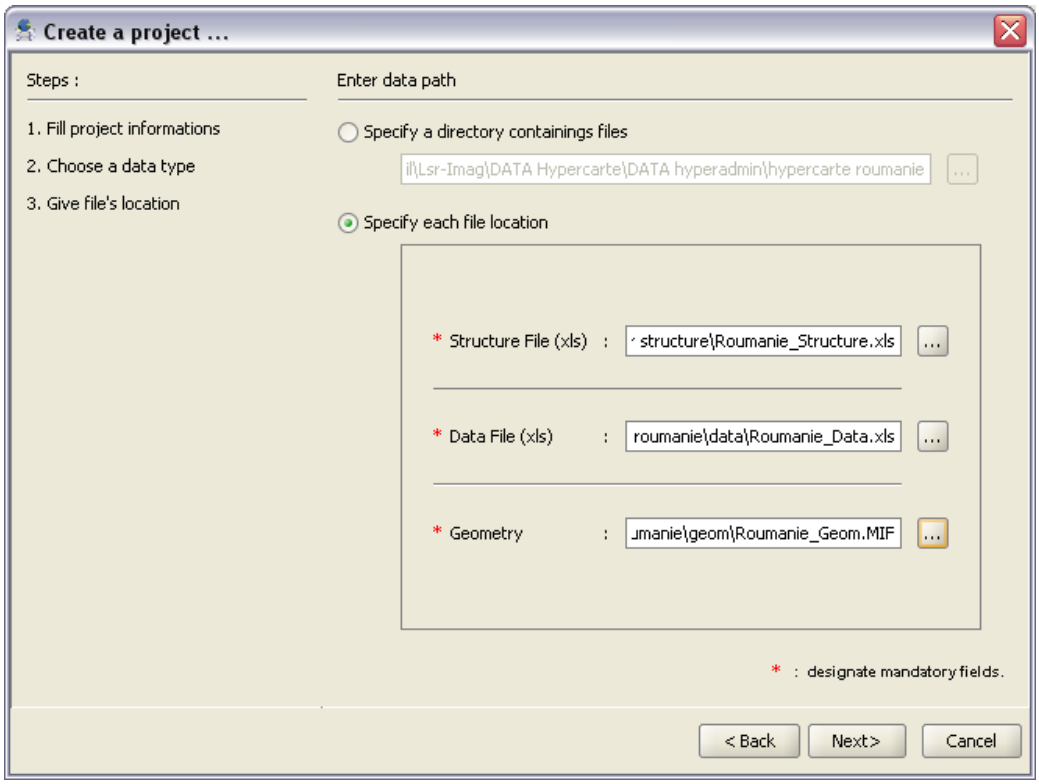


Figure 30. Troisième étape pour l'import de données avec *HyperAdmin V0.1 Alpha 2.*

Il faudrait permette à l'administrateur de construire des jeux de données progressivement : commencer par définir un jeu de données minimal, et plus tard pouvoir modifier ce jeu de données par les actions suivantes :

- Ajouter, modifier ou supprimer des maillages
- Ajouter, modifier ou supprimer des aires d'étude
- Ajouter, modifier ou supprimer des stocks
- Ajouter, modifier ou supprimer des thèmes.

Le jeu de données minimal comprend : un fond de carte, deux stocks (pour pouvoir calculer un ratio), une aire d'étude et un maillage[29].

A partir de cette demande générale, nous avons écrit en mars 2007 un cahier de spécification comprenant quelques maquettes écrites en HTML pour montrer un aperçu de notre proposition, en détaillant les opérations suivantes :

- Configuration de l'accès à la base de données,
- Connection/Déconnection de la base de données,
- Ouverture (lecture depuis la base) d'un projet,
- Création d'un projet à partir de fichiers ASCII,
- Mise en place de filtres pour la création de jeux de données restreints,
- Exportation d'un projet dans un fichier sérialisé,
- Modification d'un jeu de données.

Enfin, le prototype ne permet pas de visualiser un projet **avant** de l'enregistrer en base : en effet, le chargement des données depuis les fichiers ASCII et leur sauvegarde dans la base de données se font dans la foulée. De même, avec la structure actuelle du logiciel, toutes les modifications qui pourraient être apportées par l'utilisateur impliquent la manipulation des données de la base. Ainsi, tout retour en arrière s'avèrerait impossible. Or la possibilité d'annuler des manipulations de données est une demande des utilisateurs, qui va de paire avec la possibilité de pré-visualiser les données dans *HyperAdmin* avant leur exportation dans des fichiers sérialisés pour *HyperAtlas*.

c) Nouveauté concernant HyperAtlas

Une demande qui ne concerne pas directement le prototype existant qui n'est pas en relation avec l'ancien cahier des charges nous est soumise : il serait intéressant sur *HyperAtlas* puisse utiliser d'autre **contexte pour la déviation locale** que la contiguïté, seul choix disponible. L'idée est de permettre, par exemple, d'utiliser des voisinages fondés sur des distances temps-voiture ou temps-camion. Le projet possède quelques données sur le projet Européen, disponibles au niveau du maillage NUTS 2 uniquement. Ces données sont fournies sous la forme de matrice de distance entre unités de NUTS 2 dans un fichier ASCII, donnant le temps d'accès entre deux unités en voiture ou bien en camion. Cette nouveauté va influencer considérablement notre modèle de données, et donc notre conception de *HyperAdmin*.

En effet, contrairement aux stocks qui sont additifs, les relations de voisinage fondées sur des distances d'accès entre unités ne supportent pas le mécanisme d'agrégation. Par exemple, si l'Isère est évaluée à 4 heures de voitures du Puy de Dôme, comment déterminer la

[29] Avec un seul niveau maillage, nous ne pouvons calculer de déviation moyenne (puisqu'il n'y a alors plus d'unité supérieure englobante), mais nous pouvons adapter *HyperAtlas* pour qu'il cache cette carte.

distance entre leurs unités hiérarchiques supérieures, respectivement Rhône-Alpes et l'Auvergne ? Lyon en Rhône-Alpes est à 2 heures de voiture de Clermont-Ferrand en Auvergne, mais Grenoble en Rhône-Alpes est à 4H de voiture. Ainsi, ces matrices de distance ne sont valables que sur un niveau de maillage donné, et une aire donnée. Ces relations ne peuvent pas non plus être calculées à partir des données fournies actuellement. Nous pourrions peut-être recalculer ces matrices de distances à partir d'algorithmes connus reposant sur la modélisation des réseaux routiers. Cependant, notre application n'intègre pas la lecture de telles données. Faire ce choix nous conduirait à changer le format des données lues par *HyperAdmin* et à développer une nouvelle API de lecture des données (dont le volume est inconnu), alors qu'on nous propose une matrice de distance, et que la lecture de ce format demande simplement une adaptation légère de notre mécanisme d'import de données.

Cette demande implique d'imaginer une structure de données qui convienne pour représenter les relations de distances, et les exploiter dans *HyperAtlas*, sachant qu'on devra alors tenir compte du niveau de maillage et de l'aire d'étude.

4.1.2. Audit du prototype existant

Pour comprendre notre contribution, il est nécessaire de donner un aperçu global de l'architecture et les principes sur lesquels repose le prototype existant. Ensuite, nous précisons quelles sont les faiblesses de cette architecture au regard des besoins exprimés.

a) Architecture

Le principe retenu est le suivant : les données fournies par les géographes lors de la création d'un projet, que nous avons illustrée avec les Figure 28, Figure 29, et Figure 30, sont directement insérées dans la base de données.

La base de données (nommée *hyperadmin*) est supportée par un Système de Gestion de Base de Données (SGBD) relationnel qui est *PostgreSQL*, version 8.1. Ce serveur *PostgreSQL* présente l'avantage de s'installer sur un système d'exploitation Unix, Linux ou Windows, de façon très conviviale. L'administration et la configuration de la base se font par un module graphique, très pratique, nommé « *pgAdmin III* . Le serveur permet d'offrir des accès distants à la base de données.

De plus, le serveur est étendu avec des fonctionnalités spatiale *PostGIS*, qui permettent de manipuler des objets géométriques aisément. Par exemple, déterminer la contiguïté de deux unités territoriales (modélisées par des polygones P1 et P2) consiste simplement à invoquer une méthode *touches (Polygone P2)* sur le polygone P2, qui renvoie vrai si P1 et P2 partagent une arête commune. *PostGIS* manipule aussi des multi-polygones ou bien des formes trouées facilement. Or, le jeu de données européen par exemple comporte quelques multi-polygones (des unités territoriales regroupant plusieurs polygones non contigüs) comme la Russie avec l'enclave de Kaliningrad. Par ailleurs, les formes trouées sont surtout visibles en Allemagne, où les unités de niveau NUTS 3 s'imbriquent souvent l'une dans l'autre.

Dans cette base de données, un schéma relationnel correspondant au modèle des objets utilisé dans *HyperAtlas* a été implanté. Quelques fonctions supplémentaires ont aussi été ajoutées pour optimiser les performances des requêtes sur la base. Ces fonctions, qui se

présentent sous la forme de procédures SQL pré-compilées vont permettre, par exemple, de déterminer l'ensemble des unités contiguës à une autre unité, sans utiliser l'ancien algorithme qui n'était pas très performant.

Les données sont donc insérées en base, à partir d'un modèle objet qui n'est pas celui du paquetage *hypercarte.data*. En effet, Christophe Chabert [Chabert, 2007] a conçu un nouveau paquetage pour modéliser les données, *hypercarte.entity*. La figure 31 donne une vision d'ensemble du processus d'exploitation de la base. La lecture des fichiers textes est réalisée dans l'objet *NewProjectModel*, par l'appel de méthode *processInputs*(). Ces entités sont injectées dans la base de donnée par un ensemble de requêtes appropriées (*DBOuput.createProject()*).

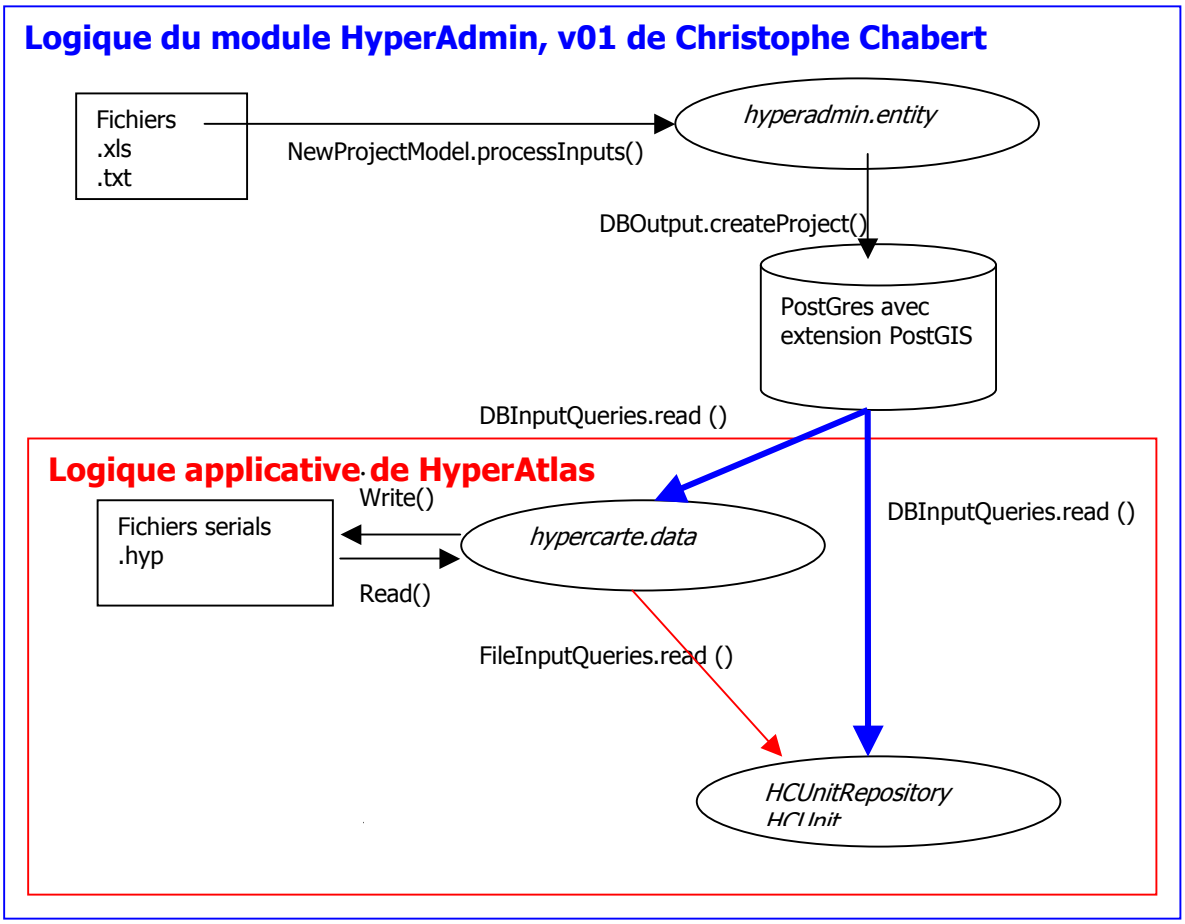


Figure 31. Logique applicative du prototype *HyperAdmin V0.1 Alpha 2.*

Après avoir été insérées en base, les données sont injectées dans deux objets utilitaires créés par Christophe Chabert, *HCUnitRepository* et *HCUnit*, qui composent une couche de transition entre le modèle des données et la couche technique manipulant ces objets (*Logic*). Ces deux objets calculent des informations telles que les relations de contiguïté, ainsi que l'agrégation des unités sur les tous niveaux hiérarchiques de l'arbre de données.

La lecture des données est modélisée par une interface, *InputQueries*, qui est implémentée ensuite de diverses manières selon les modules. Dans *HyperAdmin*, elles sont extraites de la base de données, par des requêtes qu'implémente *DBInputQueries*. Dans

HyperAtlas, les données sont lues classiquement depuis le fichier de données sérialisées, par la méthode *read()* de *FileInputQueries*. Enfin, la sérialisation des données dans des fichiers se fait avec le paquetage *hypercarte.data*.

b) Critiques

Le prototype tel que nous venons de le décrire représente une somme de travail assez considérable : le modèle de données relationnel, et les opérations de lecture et écriture en base sont une avancée pour le projet. Cependant, il présente plusieurs faiblesses.

Le premier défaut est que le code de HyperAdmin n'est pas une simple extension du code de HyperAtlas : il duplique complètement le code de HyperAtlas dans une autre paquetage. Cette duplication met en danger la maintenance de HyperAdmin : il faut en effet systématiquement reporter la résolution des bogues de HyperAtlas dans le code de HyperAdmin (et vice-versa).

Le second défaut est lié au processus d'affichage des données, tel que nous l'avons illustré avec la Figure 31 : dans HyperAdmin, la lecture des données est faite directement depuis la base, et comme aucune donnée n'est chargée en mémoire, les performances de l'application et son interactivité sont pénalisées. Toute modification du paramétrage (changement d'aire d'étude, maillage, stock, contexte de déviation) implique des lectures sur la base de données. Ne pas charger en mémoire les données donne l'avantage de pouvoir traiter des projets avec des gros volumes de données. Cependant, étant donnée le niveau de perte de réactivité de l'interface, il faut reconsidérer ce choix. Le tableau 4 illustre par exemple les temps de calcul des cartes de déviation locale, ayant pour contexte la contiguïté. A chaque modification de niveau de maillage, une requête sur la base de données est traitée, et l'utilisateur attend…

	Nb units	HyperAdmin	HyperAtlas
NUTS_0	29		15
NUTS_1	92	50 219	31
NUTS_2	280	68 872	62
NUTS_2-3	727	114 875	93
NUTS_3	1329	207 422	150

Tableau 4. Temps de calcul (ms) des cartes de déviation locale, comparés sur les divers NUTS en Europe.

Enfin, le prototype génère un jeu de données sérialisés pour HyperAtlas dans un paquetage *hypercarte.data* dont nous exposons au moins trois inconvénients :

- il ne supporte aucune évolution,
- l'algorithme de calcul des contiguïtés est gourmand en mémoire,
- les multi-polygones ne sont pas tolérés.

Précisons le premier point : aucun des objets du paquetage *hypercarte.data* ne possède d'identifiant unique de flux, ce qui empêche la modification des classes du modèle, sous peine de ne plus pouvoir désérialiser (autrement dit lire) d'ancien jeux de données. Or ce modèle de données va être enrichi afin d'autoriser l'emploi de matrices de distances, et ultérieurement

adjoindre des données descriptives aux stocks. Mais nous voulons que HyperAtlas soit encore capable de lire les anciens jeux de données. Ce n'est possible que si nous introduisons un champ *serialVersionUID* dans les objets du modèle.

Pour comprendre le deuxième point, nous décrivons le procédé de calcul de la contiguïté : chaque unité est enregistrée dans une table de hachage, dont les clés sont les points composant le contour géographique de l'unité, la liste des unités voisines partageant ce point. Ainsi, pour connaître les unités contiguës à une unité donnée, il faut parcourir cette table de hachage, et créer une liste de voisins par ajout dans un ensemble (qui ne tolère pas de doublons, ce qui occasionne des comparaisons supplémentaires). Cet algorithme est coûteux en place mémoire parce qu'il exige de stocker en mémoire la liste des points composant le contour géographique, auxquels on associe une liste de chaîne de caractères identifiant les unités partageant ce point. De plus, le contour est stocké en fait deux fois puisque la géométrie est déjà enregistrée dans un objet *java.awt.geom.Area*, pour les besoins de la couche graphique.

Enfin, le dernier point concerne la manipulation des géométries dans *HyperAtlas* : elle n'est pas adaptée à la manipulation d'unités territoriales composées de plusieurs polygones, car la géométrie des objets de type Unit est stockée sous forme d'entités de type *java.awt.geom.Area* ne permettant pas de manipuler des multi-polygones.

4.2. Proposition technique

Notre proposition comporte deux points essentiels : le premier aspect traite de la récupération du code d'*HyperAdmin*, version *V0.1 Alpha 2*, en exploitant l'existant d'*HyperAtlas* pour étendre l'interface graphique. Le second point concerne le modèle de données d'*HyperAtlas*, que nous allons modifier en fonction des nouveaux besoins apparus, ainsi que des défauts que nous avons déjà cités.

4.2.1. Ré-écriture de HyperAdmin

L'idée à terme est d'aboutir à une version d'*HyperAdmin* qui :

- traite un nouveau modèle de données qui soit évolutif,
- charge ce modèle en mémoire comme *HyperAtlas*,
- autorise des modifications, ajouts, suppressions d'entités sur ce modèle,
- sauvegarde le modèle de façon persistante dans la base,
- exporte le modèle dans des fichiers sérialisés.

Les opérations de sauvegarde persistante en base ou dans un fichier sont optionnelles. L'utilisateur qui a lu un ensemble de fichiers de données n'est pas obligé de les sauver. *HyperAdmin* est distribué aux utilisateurs autorisés comme un *HyperAtlas* étendu avec un menu pour l'administration des données.

Pour écrire ce menu et implémenter les interactions avec le modèle de données, nous récupérons une partie du travail réalisé : les opérations de lecture et d'écriture dans la base de données, ainsi que le lecture de fichiers ASCII. Etant donné que nous désirons modifier le modèle de données, le code traitant la lecture et l'écriture de fichiers sérialisés est abandonné, de même que celui qui joue l'intermédiaire entre le modèle de données et la couche métier (*HCUnit* et *HCUnitRepository*) : *FileInputQueries* doit être réécrit. Nous évitons toute

duplication de code : la partie graphique et la logique applicative relevant de l'administration des données sera déportée dans un paquetage *hypercarte.hyperadmin*, tandis que sinon, nous réutilisons le code de *HyperAtlas*, *hypercarte.hyperatlas*, pour tout ce qui concerne l'affichage des cartes et le traitement des paramètres.

Le paragraphe suivant expose nos idées concernant les interactions spécifiques qu'*HyperAdmin* propose à l'utilisateur.

a) Le menu Admin

Chaque utilisateur depuis son poste client exécutant *HyperAdmin* peut se connecter sur la base de données spatiale *hyperadmin* (locale ou distante). L'application peut sauvegarder dans un fichier plusieurs configurations de connexion, que l'utilisateur sélectionne dans l'interface graphique, afin de travailler avec une seule base durant sa session.

Ensuite, au lieu de demander le choix de la langue d'édition de ses données (comme dans la Figure 28), l'application détermine, à partir du contexte de l'utilisateur, sa préférence linguistique. Si les données ne sont pas fournies avec une description correspondant à cette préférence linguistique, alors *HyperAdmin* emploie la traduction en anglais (si elle existe), ou bien alors la seule traduction disponible. L'éventail des langues supportées pour les traductions est défini à l'avance dans le schéma de la base de données.

Le menu spécifique à *HyperAdmin* inséré dans la barre de menu de *HyperAtlas* permet de déployer progressivement des sous-menus : la sélection d'une entrée à gauche fait apparaître les choix possibles à droite (cf. figure 32)

Open data source	New connection
	Open existing connection
Open project	New project
	Open project
Export project	
Set filters	
Edit AREAS Edit ZONINGS Edit THEMES Edit STOCKS	
Commit	
Revert	

Figure 32. Proposition pour le menu *Admin*

Ces menus permettent de :

- Gérer la connexion avec la base de données,
- Ouvrir des projets qui, soit existent en base, soit sont à lire depuis des fichiers,
- Exporter un jeu de données (c'est-à-dire créer un fichier sérialisé),
- Définir un filtrage sur les données pour n'afficher qu'un sous-ensemble du projet,
- Modifier les données existantes : aires d'étude, maillages, thèmes, indicateurs,
- Sauvegarder les modifications ou les nouveaux projets dans la base,
- Annuler toutes les modifications apportées au jeu de données.

La sauvegarde des modifications sur le jeu de données doit être expressément requise par l'utilisation du menu « *commit* » en cours de session. En effet, dans cette version du logiciel, contrairement à la précédente, les modifications du jeu de données ne sont pas automatiquement enregistrées dans la base de données. En cas de sortie de *HyperAdmin* (menu *Exit*), et si des modifications sur les données sont détectées durant la session, l'application propose une confirmation de sauvegarde. Lorsque la sauvegarde est demandée, la base de données est mise à jour **uniquement** pour les valeurs modifiées, les nouvelles données, et les données supprimées. Il est possible d'annuler des modifications qui n'ont pas été encore sauvées en base par l'usage du menu « *revert* », qui permet de revenir à l'état initial (l'état suivant la dernière sauvegarde des données en base).

Toutes les opérations lentes (durée supérieure à 5 secondes) afficheront une barre de progression de tâche.

Au niveau du traitement des données, on pose l'hypothèse suivante : il est possible de travailler avec des stocks qui ne sont renseignés qu'à un certain niveau de maillage, et sur une aire donnée. Si un stock n'est pas valorisé sur une partie de l'aire d'étude sélectionnée, ou sur le maillage considéré, il ne sera pas présenté dans la liste des numérateurs ou dénominateurs possibles.
Par exemple, prenons un utilisateur qui travaille sur la France comme aire d'étude et qui connaît seulement au niveau départemental le « nombre d'inactifs ».

- S'il sélectionne le maillage de niveau communal, l'indicateur « nombre d'inactifs » disparaît.
- Sur le maillage départemental par contre, ce stock est disponible, et l'utilisateur peut toujours calculer la déviation globale, moyenne et locale.
- En sélectionnant l'Allemagne comme aire d'étude dans son projet, l'indicateur disparaît aussi.
- En sélectionnant une aire recouvrant l'Allemagne et la France, l'indicateur ne sera pas proposé.

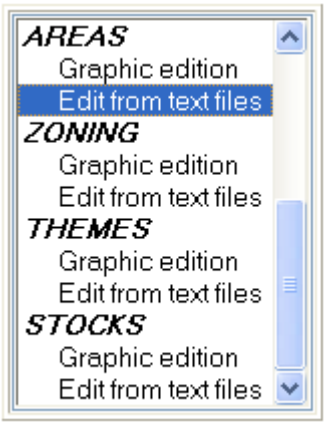
On imagine de proposer une option forçant la visualisation de cet indicateur, avec des zones grisées sur les valeurs manquantes.

b) La modification des données

Pour comprendre notre réflexion concernant le modèle de données, nous exposons nos idées concernant la modification du jeu de données, en soulignant en particulier la difficulté posée par la modification d'un maillage.

Le jeu de données peut être étendu, restreint et modifié sur différents points : les aires d'études, leur maillage, les thèmes et les données associées au projet. On procède par deux manières différentes :

- par import/modification de fichiers textes et MIF/MID (ayant le même format que ceux exportés par l'application),
- graphiquement, via la sélection de zones sur la carte, ou par l'édition dynamique de feuille Excel à l'intérieur de l'application.



L'édition des données en mode textuel se fait toujours par la modification des fichiers textes, puis en sélectionnant dans l'interface graphique le menu « *Edit from text files* », comme l'illustre la figure 33.

Figure 33. Menu de modification des jeux de données.

Dans ce cas, une boîte de dialogue pour sélectionner le répertoire contenant les fichiers ASCII définissant aire, zoning, stock, et thème s'ouvre, comme l'illustre la figure 34.

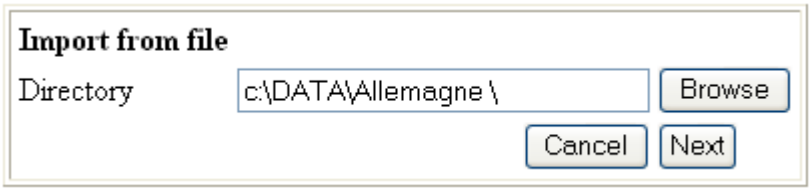


Figure 34. Dialogue pour la lecture de fichiers de données.

Puis, dans le répertoire choisi, l'utilisateur peut sélectionner les fichiers à importer (cf. figure 35). Toutes les données du fichier remplacent celles en cours dans le projet.

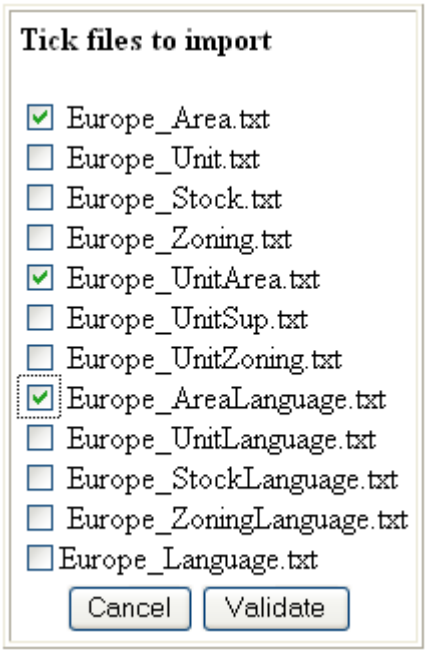


Figure 35. Sélection des fichiers ASCII à lire.

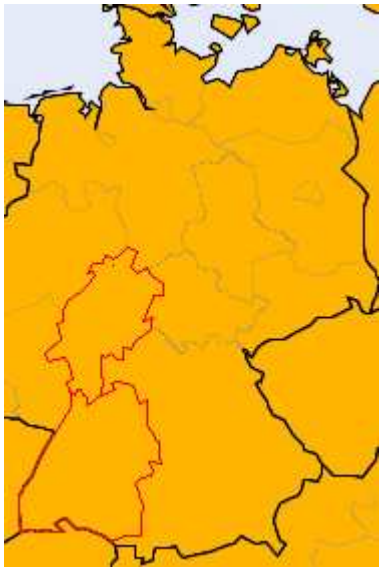
L'application effectue une suite de vérifications sur les fichiers de données et les Unités Territoriales (UT) ainsi définies:

- fichier MIF-MID : non-chevauchement des UT, rejet des doublons dans le MID ;
- contrôle de la hiérarchie des UT : on refuse l'existence de cycles dans le graphe ;
- contrôle que toutes les UT ont un contour géométrique défini.
- contrôle des maillages : une UT et son UT supérieure ne peuvent pas appartenir au même niveau de maillage ;
- contrôle des aires : il est interdit de définir une aire avec des UT de niveau hiérarchique

différent ;
- contrôle des stocks : les doublons sont interdits.
- contrôle des thèmes : les cycles dans le graphe des thèmes sont rejetés.

Chaque type d'erreur bloque la mise à jour avec un message explicite, internationalisé, et les messages sont groupés par fichiers.

c) La modification d'un maillage via l'interface



On peut ajouter, supprimer, ou modifier des maillages d'un projet à la condition qu'il reste au moins un maillage d'étude contenant une unité territoriale.
Après avoir sélectionné ZONINGS → Graphic edition dans le menu d'HyperAdmin, il est possible de sélectionner plusieurs unités dans le contexte d'étude courant, correspondant à la sélection effectuée dans le panneau Area & Zoning (cf. figure 18). Cette opération s'effectue par la sélection continue des unités avec un clic gauche de souris, en pressant simultanément la touche SHIFT du clavier. La sélection est terminée lorsque la touche SHIFT est relâchée. Les zones sélectionnées ont un contour rouge.

Figure 36. Sélection d'unités pour leur modification.

Une boîte de dialogue apparaît pour spécifier le type d'opération à réaliser sur les zonings (cf. figure 37) :
- Diviser des unités en sous-unités,
- Regrouper des unités pour fabriquer une nouvelle unité,
- Choisir le maillage auquel vont appartenir ces nouvelles unités : le maillage courant ou bien un nouveau.

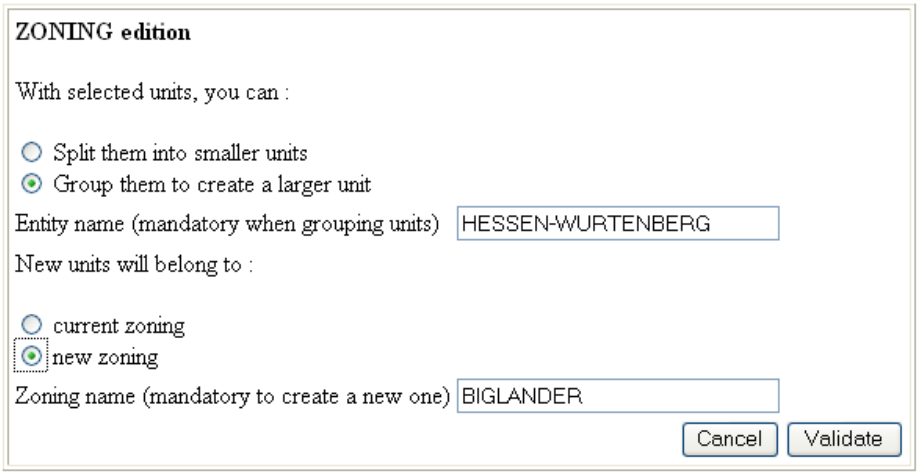


Figure 37. Dialogue pour l'édition d'un maillage.

L'opération de regroupement (*group*) de deux unités en une nouvelle unité demande de faire un choix (par exemple DE1 et DE2 appartenant à DE, et que l'on veut regrouper dans DE3) :

- soit DE3 est insérée au niveau du maillage courant LANDER : alors DE1 et DE2 sont exclues du maillage courant, et DE devient l'unité supérieure de DE3.
- soit DE3 est insérée dans un nouveau maillage de niveau supérieur BIGLANDER : DE devient l'unité supérieure de DE3.

Dans tous les cas, DE3 devient l'unité supérieure de DE1 et DE2. La nouvelle unité territoriale créée se voit attribuer un identifiant unique, et son nom traduit est celui présent dans le formulaire.

L'opération de division en plus petites unités (*split*) implique l'import de nouveaux contours géographiques pour les nouvelles sous-unités. L'application propose alors un tableur Excel intégré composé de trois colonnes : la première pour définir les identifiants texte des nouvelles unités, la deuxième pour leur donner un nom (dans la langue utilisée), la troisième pour leur définir des contours géométriques sous la forme d'une suite de points (cf. tableau 5). Le système de coordonnées utilisé pour la géométrie est supposé être identique à celui utilisé dans le projet.

UT-ID	LANGUAGE NAME	GEOMETRIE
FR714	Isère	POLYGON((-764623.617 -414408.063,-755610.762 -420480.945,-743628.756 -409488.744, (..)"
FR715	Loire	POLYGON((-849520.677 -350423.094,-844759.074 -363310.899,-821895.612 -367962.612, (..)"
FR716	Rhône	POLYGON((-789975.963 -436260.591,-785985.489 -443424.087,-796030.905 -449967.999, (..)"
FR711	Ain	…
FR712	Ardèche	…
FR713	Drôme	…
FR717	Savoie	…
FR718	Haute-Savoie	…

Tableau 5. Exemple de tableur Excel pour la saisie de nouvelles unités.

L'application vérifie alors les conditions suivantes avant d'accepter la modification :

- les nouvelles unités ne se chevauchent pas
- elles recouvrent entièrement la zone sélectionnée
- elles sont incluses dans la zone sélectionnée
- leur identifiant est unique

Ensuite l'application doit opérer certains changements sur la hiérarchie des unités. Par exemple, dans le maillage LANDER, DE2 qui appartient à DE vient d'être divisée en DE21 et DE22.

- soit DE21 et DE22 appartiennent désormais aussi au maillage courant LANDER : alors DE2 n'appartient plus à ce maillage, et DE devient l'unité supérieure de DE21 et DE22.
- Soit DE21 et DE22 font partie d'un nouveau maillage LITTLELANDER de rang inférieur à celui de LANDER. L'unité supérieure de DE21 et DE22 est DE.

Tant qu'aucune aire d'étude couvrant strictement toutes les unités du nouveau maillage n'est définie, l'application cache ce nouveau maillage. Cette description d'une opération de modification de maillage montre que, pour offrir de telles possibilités, l'application doit procéder à de nombreuses vérifications et automatiser certaines tâches pour l'utilisateur. Or, pour cela, un modèle de données souple et robuste est nécessaire.

4.2.2. Evolution du modèle de données

Nous avons vu que le modèle de données implémenté dans *hypercarte.data* présente certaines faiblesses, auxquelles nous souhaiterions remédier.

Pour le manque d'évolutivité du modèle, la solution est assez simple : nous allons implémenter la sérialisation des objets en suivant les recommandations du chapitre 10 de Joshua Bloch, [Bloch, 2002], avec en particulier l'ajout de numéros de série UID (*serialVersionUID*) comme attributs de chaque classe d'objet sérialisé. Ainsi, les jeux de données supporteront désormais des modifications de leur structure, et ce sera au développeur d'assurer la compatibilité ascendante, en adaptant les méthodes de *readObject(…)* et *writeObject(…).*

En ceci concerne l'algorithme de calcul des contiguïtés, ainsi que pour le support des multi-polygones, la solution commune se trouve dans l'utilisation de bibliothèques spatiales. Nous avons vu que *PostgreSQL* propose *PostGIS*, et des procédures pratiques et rapides pour le calcul des contiguïtés. De plus, *PostGIS* manipule des objets géométriques assez variés, dont les multi-polygones, ou encore les surfaces trouées. Cependant, nous avons remarqué que JTS[30] (*Java Topology Suite*) est une bibliothèque de fonctions spatiales très couramment utilisée dans le développement de SIG libres [Plumejeaud, 2005], car elle présente l'avantage d'être sous licence LGPL, et se conforme aux spécifications publiées par l'OpenGIS. Cette bibliothèque, écrite en Java, offre la manipulation de nombreuses formes géométriques et des algorithmes de test d'inclusion, de contiguïté, etc. Nous avons là un choix à faire, qui nécessite quelques tests. En effet, lorsque nous extrayons les données de la base, la traduction en objet de type PostGIS est directe, alors que pour les objets JTS, cela demande un peu plus de travail. Mais, si le calcul des contiguïtés est plus rapide avec JTS, ou la place mémoire occupée par des objets de type JTS est moindre, ce travail serait rentabilisé.

Notre champ d'action sur le modèle de données ne se limite pas à améliorer l'existant : nous devons aussi l'aider à supporter les nouvelles fonctionnalités. La plus prioritaire (de notre point de vue) est de pouvoir exploiter des matrices de distance, pour proposer d'autres contextes pour la déviation locale que la contiguïté entre unités. Nous voulons aussi pouvoir manipuler la hiérarchie des unités, et leur appartenance à des maillages ou aire d'étude plus facilement que dans le modèle actuel. Nous souhaitons aussi augmenter les performances du modèle.

a) Introduction de matrice de relations valuées

Les contextes pour la déviation locale utilisent des relations « horizontales » entre objets, dans le sens où des unités de même niveau hiérarchique sont mises en relation, et dans notre cas nous étudions les relations de proximité. L'analyse spatiale postule alors que les

[30] http://www.vividsolutions.com/jts/jtshome.htm

caractéristiques d'un lieu dépendent des relations de proximité de ce lieu par rapport à d'autres lieux [Pumain et al., 2004]. La proximité est évaluée par la **distance**, qui est une notion géographique fondamentale. Cette notion d'espacement ne respecte pas forcément les propriétés d'une distance mathématique.
La distance mathématique entre deux points A et B est une mesure toujours positive, d(A,B)>0, qui

- est nulle seulement si A est confondu avec B : $d(A,B) = 0 \Leftrightarrow A = B$;
- vérifie l'inégalité triangulaire entre 3 points A, B, C : $d(A,C) <= d(A,B) + d(B,C)$;
- est symétrique : $d(A,B) = d(B,A)$.

Or les relations de distance entre unités géographique ne sont pas forcément symétriques, et par exemple, le temps mis par un routier pour atteindre Grenoble en partant de Gênes n'est pas identique au temps de retour, en raison du relief et d'un réseau routier qui impose plus d'arrêts dans un sens que dans l'autre. Ceci tient au fait que l'espace géographique n'est pas isotrope. En fait, ce sont des mesures d'éloignement, que par commodité on continue d'appeler distance. Et la contiguïté est un cas particulier de cette mesure d'éloignement : entre deux unités A et B, la mesure vaut 1 si elles se touchent, 0 sinon.

L'idée qui émerge de cette analyse est qu'en fait, une seule structure de données suffit à définir des voisinages à partir de matrices de distance, ou bien à partir du calcul des contiguïtés. En effet, la contiguïté se modélise aussi sous la forme d'une matrice, dont les entrées valent 1 si deux unités partagent une frontière commune, et 0 autrement. Une matrice de distance donne une mesure d'espacement, que ce soit en temps ou en kilomètres, entre deux unités. Elle n'est pas symétrique, contrairement à la matrice de contiguïté. Nous pouvons donc :

- précalculer les relations de contiguïté et les conserver dans une matrice de distance,
- lire des matrices de distance directement depuis des fichiers de données.

Toutefois, pour définir un voisinage à partir d'une matrice de distance, il nous manque encore quelque chose de fondamental : un seuil numérique et un opérateur de comparaison pour ce seuil. En effet, le seuil détermine à partir de quelle distance nous considérons deux unités comme voisines. L'opérateur de comparaison donne le type de test qu'il faut effectuer entre la valeur d'espacement et le seuil. Dans le cas de la contiguïté, l'opérateur est « égal », tandis que le seuil est 1. Pour les distances temps-voiture, l'éventail des seuils est infini, mais en général nous utilisons l'opérateur « inférieur » : deux villes sont voisines si elles sont à moins de 2 heures de route par voiture par exemple. Nous proposons l'ensemble des opérateurs de comparaison suivant : $\{ <, \leq, =, \geq, > \}$.
En résumé, un voisinage est un triplet (matrice de distance, seuil numérique, opérateur de comparaison).

Voisinage = matrice de distance + seuil numérique + un opérateur de comparaison

Les matrices de distance, que nous appelons plutôt **matrices de relations valuées**, afin de bien généraliser la notion d'espacement, doivent être introduites dans le modèle. Nous envisageons d'utiliser une structure de hachage, dont les clés sont les identifiants de toutes les unités, et les entrées une liste de couples (identifiant d'unité, distance). Ainsi, à partir d'une unique matrice de distance, il est possible de définir une infinité de voisinages puisque le choix de seuils est infini, sans dupliquer les données. Par exemple, à partir d'une matrice de

distance temps-voiture entre unités, on définit trois voisinage si l'on veut : voisinage à 2 heures de route, 3 heures de route, 6 heures de route.

b) Utilisation d'un patron « Poids-Mouche »

Nous observons que l'ancien modèle de donnée utilise des **identifiants** textuels pour chaque entité (Unité, Aire d'étude, Maillage, Stock), ces identifiants sont ceux attribués par les fournisseurs de données au format ASCII. Un identifiant unique pour chaque unité est nécessaire, pour éviter de dupliquer les données. En effet, les relations d'interdépendance entre entités demandent parfois, pour des raisons pratiques, de répéter les informations. Par exemple, une aire d'étude est composée d'un ensemble d'unités, et une unité appartient à un ensemble d'aire d'étude. Lorsque l'on dessine une aire d'étude, nous sélectionnons l'objet aire d'étude, à partir duquel nous listons l'ensemble des unités le composant. Mais lorsque l'on souhaite connaître à quelle aire appartient une unité, nous ne re-parcourons pas l'ensemble des aires d'études pour déterminer si l'unité y est incluse. L'information est portée par l'unité elle-même qui liste par leurs identifiants les aires auxquelles elle appartient.

Cet identifiant pourrait cependant être optimisé puisqu'il sert souvent aux comparaisons d'entité : à la place d'une chaîne de caractères, nous envisageons d'attribuer un identifiant de type entier, par l'emploi d'une fabrique d'index. D'autre part, en base de données, les entités possèdent déjà un identifiant de type entier. Ainsi, lorsque les données sont lues depuis des fichiers ASCII, l'application attribue des identifiants entiers aux entités, et lorsque les données sont sauvegardées, ou lues dans la base, nous utilisons l'identifiant que génère la base.

D'autre part, nous sommes tentés par l'emploi du patron « Poids-Mouche » [Gamma et al, 1999], dont l'objet central serait l'Unité, avec un identifiant unique généré par la fabrique d'unité. En effet, les principales informations sont portées par l'entité Unité : le contour géométrique, la valeur des stocks, sa description. Le reste semble contextuel : l'appartenance à un zoning, ou bien à une aire d'étude, la relation de voisinage avec d'autres entités, la hiérarchie qui indique l'unité immédiatement supérieure ou bien l'ensemble des unités qui la composent. On aimerait, en effet, manipuler simplement des relations unité-maillage, unité-aire, unité-unité afin de mettre facilement en œuvre les modifications sur le jeu de données. Cependant, à cause des opérations d'addition de stocks et d'agrégation des contours géométriques sur les différents niveaux hiérarchiques, la modification du niveau de zoning ou le changement de composition d'une unité modifie la valeur des stocks, et ses contours géométriques, qui ne sont donc pas des données intrinsèques. Le patron « Poids-Mouche » ne convient donc pas tout à fait à nos besoins.

4.3. Réalisation

Nous exposons ici l'ensemble de nos réalisations, qui correspondent globalement à notre proposition technique.

4.3.1. Les étapes du développement

Le développement a suivi un cycle itératif, procédant par étapes pour la récupération progressive du code existant. Nous avons modifié le modèle de données avant de nous

consacrer à l'implantation d'un menu HyperAdmin dans l'interface de HyperAtlas. En effet, la modification du modèle impactait d'abord HyperAtlas et il fallait adapter la couche métier pour qu'elle puisse exploiter le nouveau modèle.

Lorsque cette première étape est franchie le 24 juillet 2006, (cf. figure 38), nous sommes alors à même de produire et d'exploiter des données dans le format évolutif désiré pour HyperAtlas, modèle regroupé dans le paquetage *hypercarte.hyperatlas.serials*. Par un petit programme nous extrayons les trois projets existant alors dans la base de données : Europe_ESPON, Cameroun et Rhône-Alpes (avec seulement une dizaine de variables) pour les sérialiser dans le nouveau modèle. Toute la partie concernant la sérialisation/désérialisation des données est commandée depuis *hypercarte.hyperatlas.serials.DataSerialver*, qui délègue aux entités du modèle le soin de réécrire leur propres méthodes d'écriture et lecture de flux.
L'extraction de données se fait à partir de la classe *hypercarte.hyperatlas.io. SerialInputQueries* qui implémente l'interface *InputQueries* des entrées pour *HyperAtlas.*

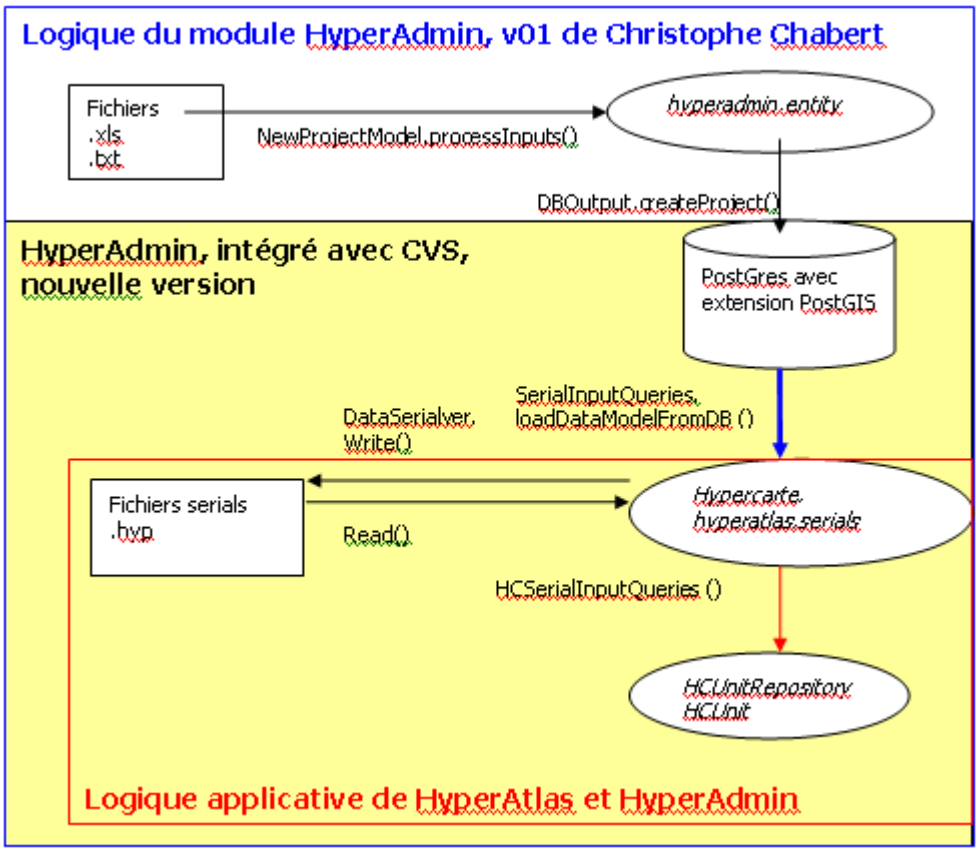


Figure 38. Première phase : exploitation d'un nouveau modèle de données dans HyperAtlas.

L'étape suivante et finale consiste à implémenter ce qui concerne l'insertion de données dans la base, à partir de données conservées en mémoire, ainsi qu'à lire le modèle de données depuis les fichiers ASCII. Nous regroupons ces classes dans le paquetage d'entrée-sortie de *HyperAdmin*. La lecture des fichiers de données est faite à partir de la classe *hypercarte.hyperadmin.io.UserDataSource*, modifiée pour « remplir » *hypercarte.hyperatlas.serials*. Tandis que la lecture et l'écriture des données dans la base sont

déléguées à *hypercarte.hyperadmin.io HASerialInputQueries*, qui exploite du code existant dans le prototype : (DBOutput.write() et DBInputQueries.read()). Cette étape est terminée le 20 décembre 2006, et son schéma de fonctionnement est illustré par la figure 39.

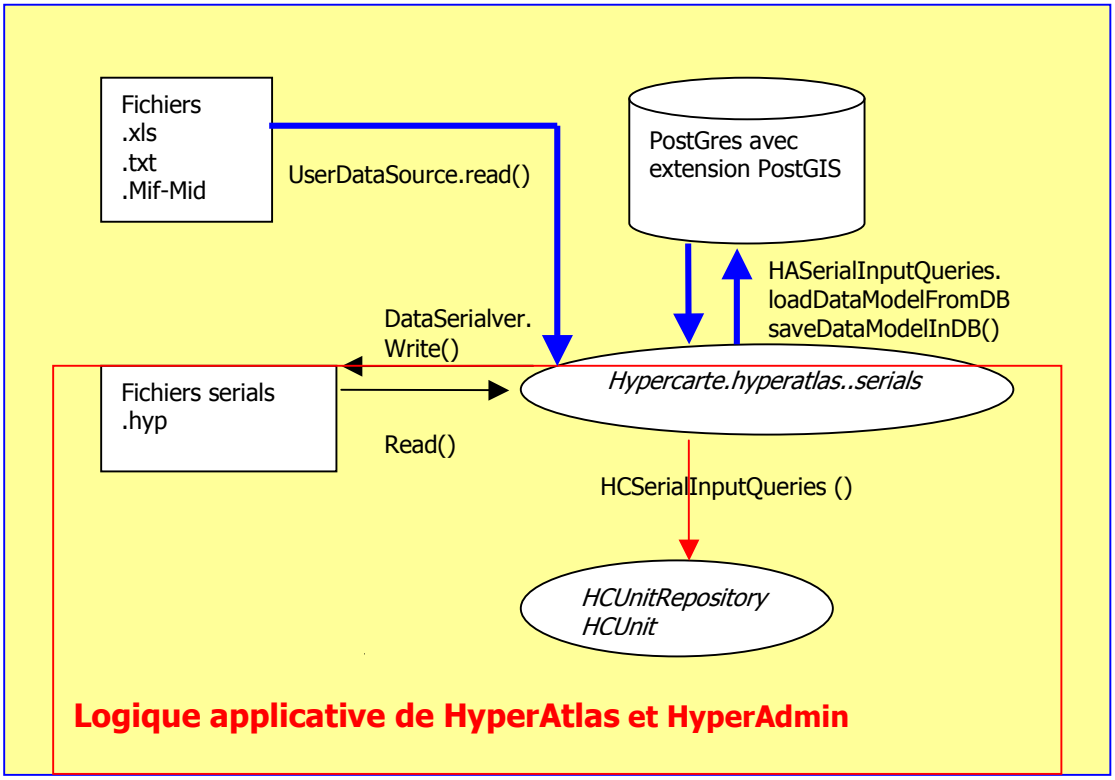


Figure 39. Seconde phase de développement : écriture des entrées-sorties sur la base de données.

4.3.2. Le modèle de données

L'implémentation d'un nouveau modèle de données consiste à ajouter des matrices de relations valuées, mais aussi à prévoir un mécanisme pour proposer au moins la contiguïté comme relation de proximité, pour ne pas perde ce choix présent dans HyperAtlas. De plus, nous nous attachons à rendre ce modèle évolutif (en suivant les recommandations concernant la sérialisation de Joshua Bloch [Bloch, 2001]) et en articulant clairement les différentes relations existant entre entités dans le modèle.

a) Les voisinages

Les relations de distance entre unités sont stockées sous forme de matrice en mémoire, considérée comme non symétriques, et non creuses même si, dans le cas de la contiguïté, on sait que très peu d'unités seront voisines (donc à une distance de 1). La structure de données choisie pour enregistrer ces relations est une table de hachage, nommée *Contiguity*, dont les clefs sont des copies légères des unités (juste leur identifiant), de type *LightUnitImpl*, et les entrées correspondantes une liste d'objets de type *DistanceUnitImpl*, associant un identifiant d'unité à sa distance à l'unité de la clef. A cette matrice de distance nous associons un seuil et un opérateur de comparaison pour définir un voisinage : c'est l'utilité de *Neighbourhood*. La

matrice de distance définie aussi des domaines de validité : aire et maillage, par l'intermédiaire de listes d'identifiants de maillages, et d'aires.

Pour rendre persistants ces voisinages et matrices de distance dans la base de données, des relations doivent être implantées dans le schéma *hyperadmin*. A cette fin, nous créons les tables *Contiguity* et *Neighbourhood*. Une petite optimisation pour la place occupée en base est faite pour les matrices de distance concernant la contiguïté : la relation étant symétrique, nous ne stockons la relation entre deux unités qu'une seule fois. *Contiguity* conserve les relations valuées entre unités (la distance, par exemple), qui sont définies par projet. Cette entité est traduite dans les langues disponibles dans la table *Contiguity_description*.

Contiguity	Attribut	Type	Contrainte sur l'attribut
	Id	Entier	PK[31]
	Project_id[32]	Entier	FK[33] on project(Id)
	Code	Texte	obligatoire (valeur nulle refusée)

Tableau 6. Table définissant les matrices de distance (Contiguity).

tu_contiguity	Attribut	Type	Contrainte sur l'attribut
	contiguity_id	Entier	FK on contiguity (id)
	tu_id1	entier	FK on unit (id)
	tu_id2	entier	FK on unit (id)
	distance	Flottant sur 8 bits	obligatoire (valeur nulle refusée)

Tableau 7. Table conservant les relations (unité-unité-distance) pour chaque type de distance défini.

La relation valuée (de distance par exemple) est valable sur une aire d'étude donnée, sur un certain niveau de maillage. Nous définissons donc deux relations « contiguity-zoning » et « contiguity-area » pour noter les domaines de validité des relations de distance.

Contiguity_area	Attribut	Type	Contrainte sur l'attribut
	contiguity_id	Entier	FK on contiguity (id)
	area_id	entier	FK on area (id)

Tableau 8. Table définissant le domaine de validité des relations de distance sur les aires.

Contiguity_zoning	Attribut	Type	Contrainte sur l'attribut
	contiguity_id	Entier	FK on contiguity (id)
	zoning_id	entier	FK on zoning (id)

Tableau 9. Table définissant le domaine de validité des relations de distance sur les maillages.

Enfin, on définit un voisinage (*Neighbourhood*) comme le triplet (relation valuée, seuil limite, opérateur de comparaison). En fonction du seuil et de l'opérateur de

[31] PK : Primary Key ; clé principale identifiant unique de l'entrée.
[32] Note : nous surlignons en bleu les n-uplets d'attributs constituant une clé d'entrée unique dans la table.
[33] Foreign Key ; référence sur l'entrée d'une autre table par un certain attribut – syntaxe : FK on table(attribut).

comparaison, deux unités sont considérées comme voisines ou non. L'entité est traduite dans les langues disponibles (*Neighbourhood_description*).

Neighbourhood	Attribut	Type	Contrainte sur l'attribut
	Id	Entier	PK
	Project_id	Entier	FK on project(Id)
	code	Texte	obligatoire (valeur nulle refusée)
	contiguity_id	Entier	FK on contiguity (id)
	distance	Flottant sur 8 bits	obligatoire (valeur nulle refusée)
	comparator	String	soit <, >, ==, <=, >=

Tableau 10. Table définissant les voisinages d'un projet.

b) Calcul des contiguïtés

La question se pose d'utiliser JTS ou bien PostGIS pour implémenter les contours géographiques des unités. En effet, nous souhaitons nous reposer sur les fonctionnalités topologiques offertes par ces deux bibliothèques pour le calcul des contiguïtés. La géométrie qui est conservée dans la base de données est de type PostGIS, et si nous employons JTS en mémoire il faut effectuer un travail de conversion de type lors de l'extraction et insertion des données en base. La question que nous nous posons est de savoir si pour effectuer toute opération topologique à l'avenir il faut être connecté sur la base ? Ce serait contradictoire avec notre objectif de pouvoir modifier des jeux de données sérialisés chargés en mémoire uniquement. Pour décider, nous effectuons des tests comparatifs de performance pour le calcul des contiguïtés, avec des géométries écrites chacune des bibliothèques. Les tests utilisent le jeu de données « Europe », et pour chaque niveau de l'arbre de donnée (donc sur chaque niveau du maillage NUTS) nous mesurons les temps de calcul. La figure 40 présente les résultats.

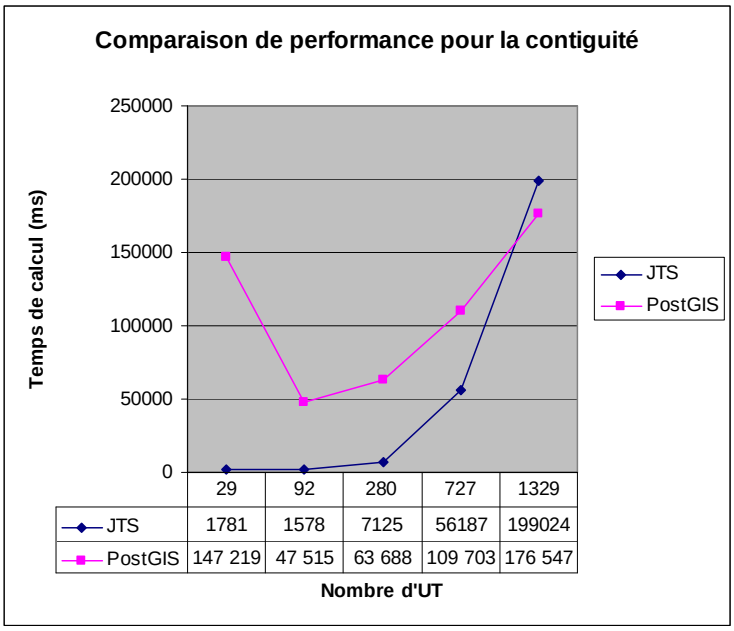


Figure 40. Comparaison des temps de calcul de la contiguïté entre JTS et PostGIS.

Nous constatons alors que sur une petite quantité d'unités, JTS est largement plus performant que PostGIS, mais que au delà de 1000 unités environ, (les contours des unités comprennent en moyenne 13 points), la tendance s'inverse et PostGIS devient plus rapide. Surtout, le calcul sur un grand nombre d'unités avec JTS utilise une quantité importante de mémoire. Le jeu de données « Europe », est trop petit pour vérifier le seuil au-delà duquel la JVM manquera de mémoire, mais on imagine que sur un jeu de données communales avec 36000 communes, JTS ne fonctionnera pas.

En conséquence, nous prenons la décision d'utiliser PostGIS lorsque le nombre d'unités est supérieur à 1000. Ceci impose d'être connecté à la base et de travailler sur des unités déjà enregistrées en base. Mais le gain en temps de calcul est vraiment considérable, et il pourrait même être fortement augmenté en employant des index géométriques sur les tables de géométrie. En dessous de 1000 unités, nous continuons de travailler avec JTS. En effet, à l'avenir, l'utilisateur pourrait avoir la possibilité de travailler sur des parties du maillage comprenant peu d'unités, et dans ce cas, JTS convient.

c) Les entités dans le modèle

Le patron Poids-mouche ne peut être employé. Mais nous concevons tout de même un modèle (schématisé par figure 41) dérivant de cette idée, où nous modélisons chaque relation avec l'unité dans une classe à part :

- *AreaSet* pour les relations aire-unité
- *ZoningSet* pour les relations maillage-unité
- *HierarchyRelations* pour les relations hiérarchiques entre unités : unité-unité
- *NeighbourhoodSet* pour les relations de voisinage entre unités : (matrice de distance, seuil, comparateur)

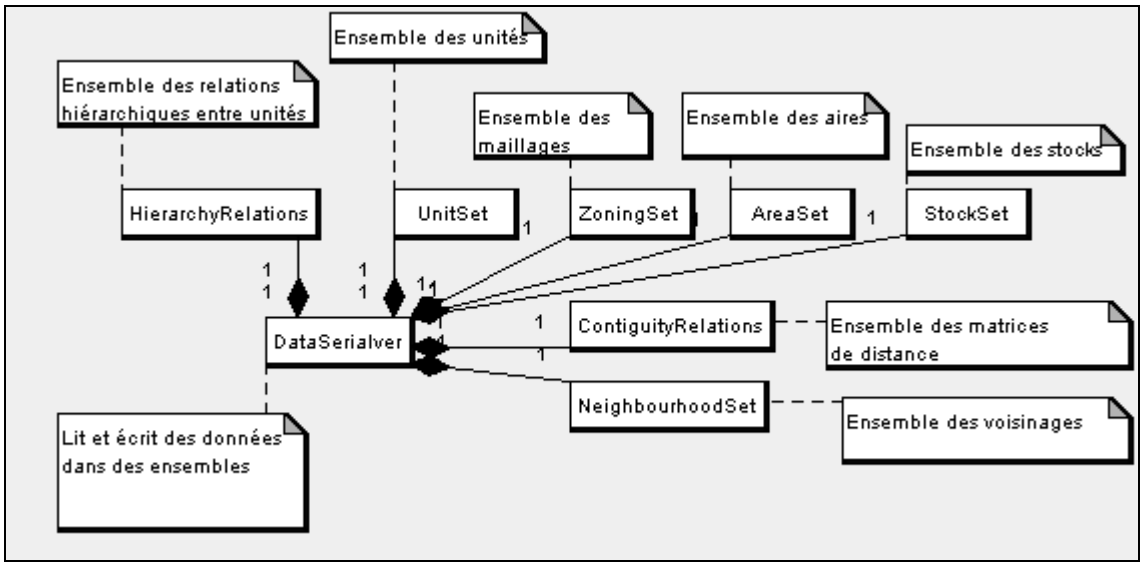


Figure 41. Diagramme de composition du modèle.

Tout le modèle est regroupé dans un paquetage, *hypercarte.hyperatlas.serials*, qui appartient à *HyperAtlas*. Nous ne plaçons pas ces données à un niveau supérieur, car elles

sont spécifiques pour *HyperAtlas*, ni sous *HyperAdmin* car alors la compilation d'*HyperAtlas* aurait une dépendance sur *HyperAdmin*.

Les entités unités, maillages, aires d'étude, stocks, voisinages et matrices de distance sont destinées à être sérialisées avec leur identifiant entier, et leur description dans différentes langues. D'autre part, ces entités sont susceptibles d'être créées, modifiées, supprimées du jeu de données, et dans le but de pouvoir faire une mise à jour de la base de données sélective, nous leur attribuons aussi trois champs booléens : *_toInsert*, *_toUpdate*, *_toDelete*. Cette représentation commune est modélisée dans un objet de type *SerialElement*, qui est spécialisé par ces entités, (cf. figure 42).

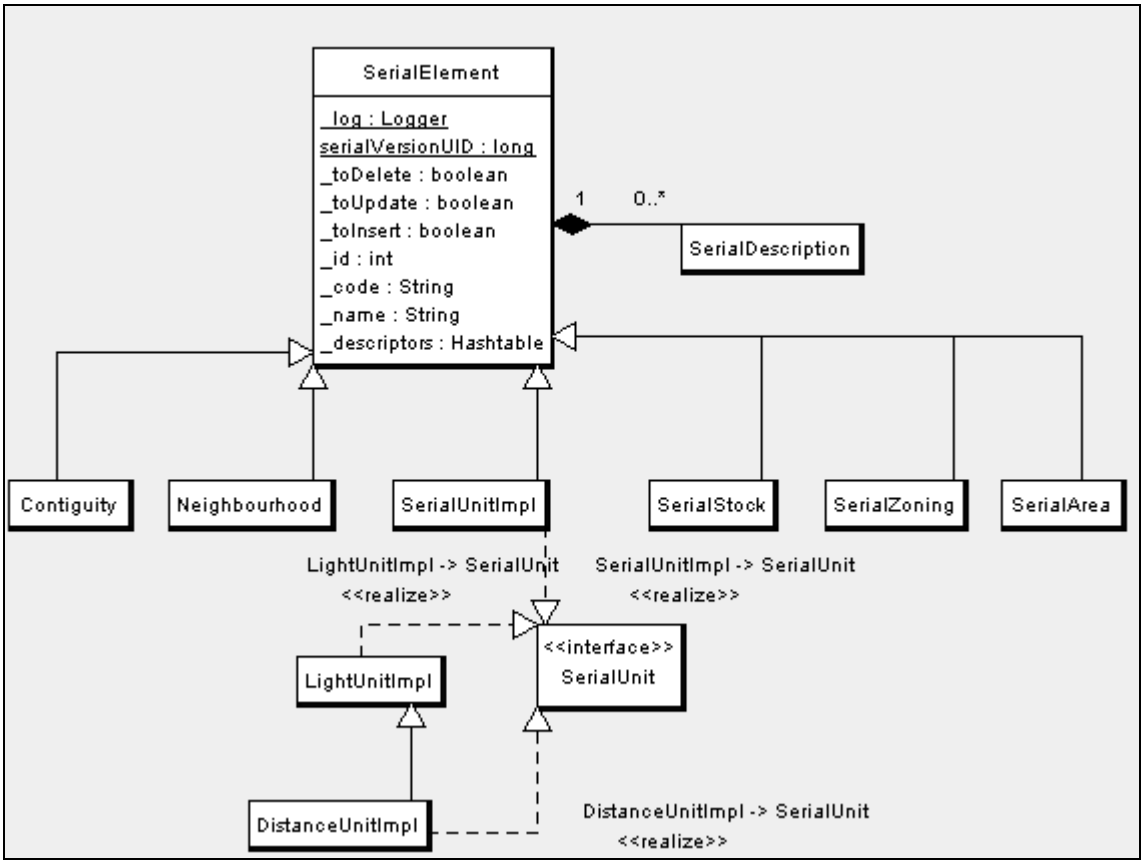


Figure 42. Diagramme de spécialisation des entités du modèle.

Remarquons l'usage des entités *LightUnitImpl* et de sa spécialisation en *DistanceUnitImpl* qui sont en fait des copies légères de *SerialUnitImpl*, destinées à être manipulées dans les ensembles d'aires, de maillages, pour la description des hiérarchies ou des relations de voisinage. Elles contiennent l'information minimum pour identifier une unité, sans géométrie, ni descriptions, ni stocks, etc.

4.3.3. Les scénarii d'utilisation

Nous présentons ici un mode d'emploi de ce module, avec les différentes possibilités qui sont effectivement offertes à l'utilisateur, sans être exhaustif toutefois.

Le menu « admin » que nous avons programmé et ajouté dans la barre des menus de HyperAtlas (durant le mois de janvier) propose les opérations décrites par le tableau 11. Elles concernent principalement l'acquisition de données. La modification des données existantes n'est pas encore implémentée.

Menu Admin		
Menu principal	**Sous-menu**	**Signification**
Database connection	**Define a connection**	Ouvre un dialogue de spécification de connexion sur une base de données: serveur, port, nom de la connexion. Les connections spécifiées sont ajoutés à un fichier XML config/connexion.xml
	Connect	Ouvre un dialogue pour ouvrir une des connexions spécifiées. On demande le login et le mot de passe. Ce menu est désactivé lorsque l'application est connectée sur un serveur de base de données
	Disconnect	Déconnecte l'application du serveur de base de données. Tant que l'application n'est pas connectée, ce menu est désactivé.
Open Project	**Read project from files**	Ouvre un dialogue permettant de créer un nouveau projet quasiment vide : une aire, un maillage et une géométrie.
	Import project from DB	Ouvre un dialogue permettant de sélectionner le projet à lire depuis la base de données. Le projet est ensuite sérialisé, et affiché dans HyperAdmin
Neighbourhood	**Create Neighbourhood**	Créer un voisinage - à partir de fichiers de données - en calculant la contiguïté simple - à partir de matrices de distance existante
Export project	Ce menu ouvre un dialogue permettant de choisir l'emplacement et le nom du fichier dans lequel les données seront sérialisées. Le fichier produit, d'extension .hyp, pourra servir ensuite à HyperAtlas ou bien HyperSmooth	
Commit	Via ce menu, l'utilisateur peut sauver un projet et les modifications apportées dans sa totalité dans la base de données. Il n'est actif que lorsque l'application est connectée.	

Tableau 11. Présentation du menu Admin et des opérations disponibles.

a) Créer des voisinages avec HyperAdmin et les exploiter dans HyperAtlas

Ce scénario est destiné à enrichir un projet existant avec des nouveaux contextes pour la déviation locale, définis dans des fichiers de données. Le projet est ensuite exporté dans des fichiers sérialisés pour être exploités par HyperAtlas. Les figures suivantes illustrent les étapes du processus.

Il faut d'abord récupérer le projet dans la base de donnée. Ce qui nécessite de se connecter (cf. figure 43), et il faut pour cela s'authentifier avec un identifiant/mot de passe (cf. figure 44).

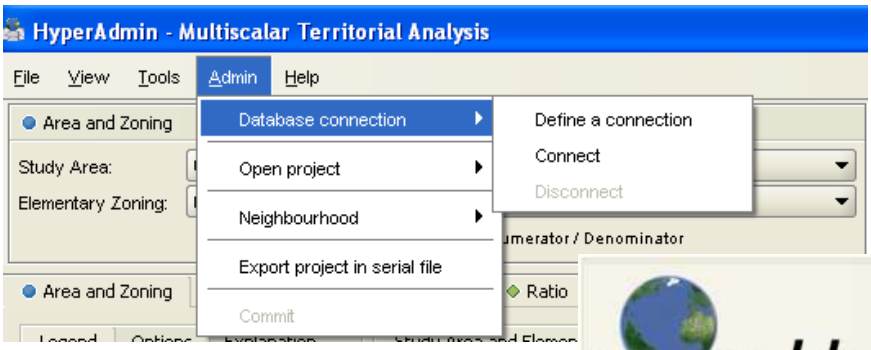


Figure 43. Menu Admin : choisir de se connecter sur la base *hyperadmin*.

Figure 44. S'authentifier et choisir la connection.

Ensuite, il est nécessaire d'ouvrir un projet par import d'un projet existant en base (cf. figure 45). Un dialogue permet de sélectionner le projet et lorsque l'import est achevé, un message d'information apparaît (cf. figure 46).

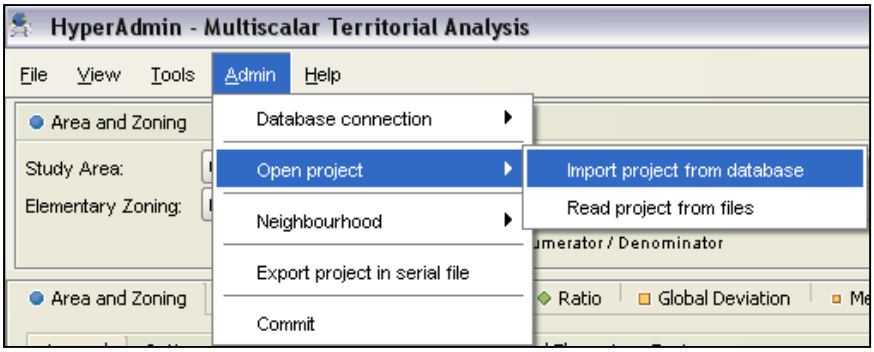


Figure 45. Menu Admin : importer un projet depuis la base *hyperadmin*.

Figure 46. Etre informé en fin de lecture que le projet sélectionné est chargé.

Ensuite, il est possible de spécifier un nouveau voisinage (cf. figure 47), ce qui ouvre un dialogue permettant de spécifier la source de la matrice de distance (à lire dans un fichier ou bien utiliser une matrice existante), le seuil et le comparateur, (cf. figure 48).

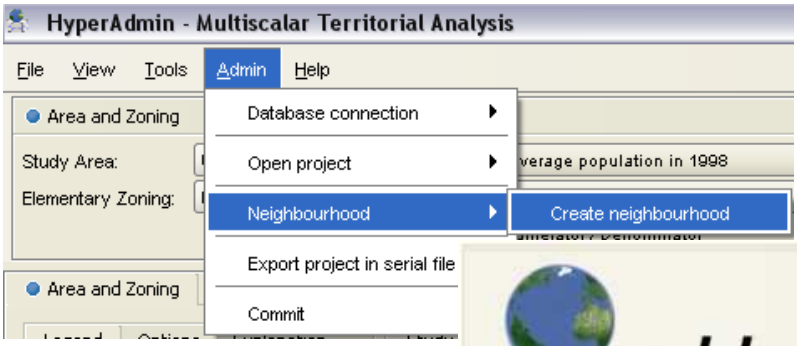


Figure 47. Menu Admin : choisir de créer un nouveau voisinage.

Figure 48. Sélectionner le mode de spécification du voisinage : à partir de fichiers de données, ou bien d'une matrice de distance existante.

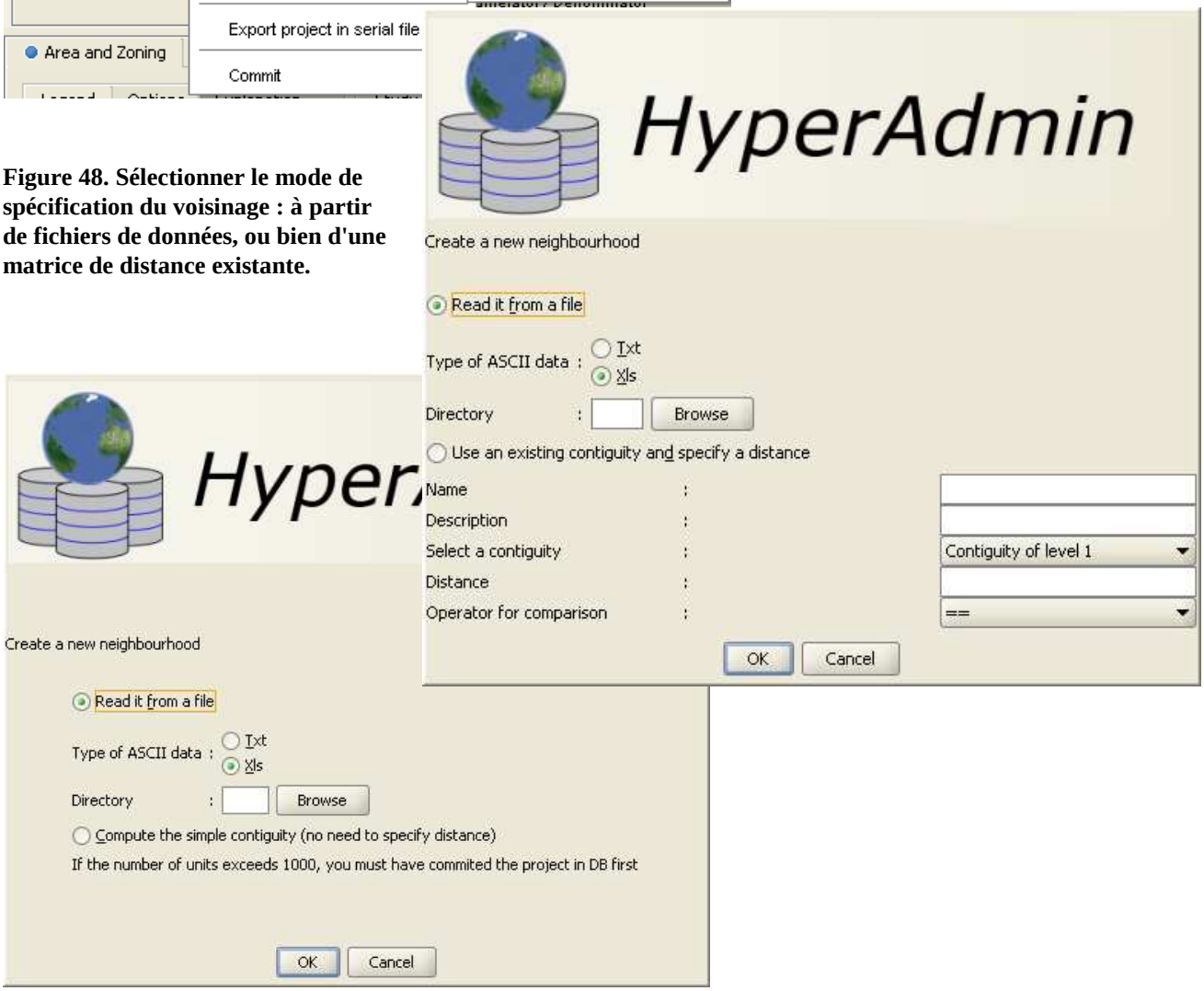


Figure 49. Choix entre des voisinages définis dans des fichiers, ou le calcul de la contiguité.

Dans le cas où il n'existe aucune relation de distance, le dialogue propose aussi de calculer la contiguïté (cf. figure 49), et précise que si le nombre d'unités est supérieur à 1000, il est nécessaire que l'utilisateur se connecte et enregistre le projet dans la base de données.

Lorsque ces étapes ont été réalisées, des nouveaux voisinages existent et peuvent être choisis en tant que contexte pour la déviation locale. On observe alors que les voisins varient

en fonction de ce contexte. Par exemple, lorsque l'on considère la contiguïté comme relation de proximité, la région de l'Irlande du sud, « Southern and Eastern », possède la région du milieu « Border, Midland and Western » pour voisine, (cf. figure 50).

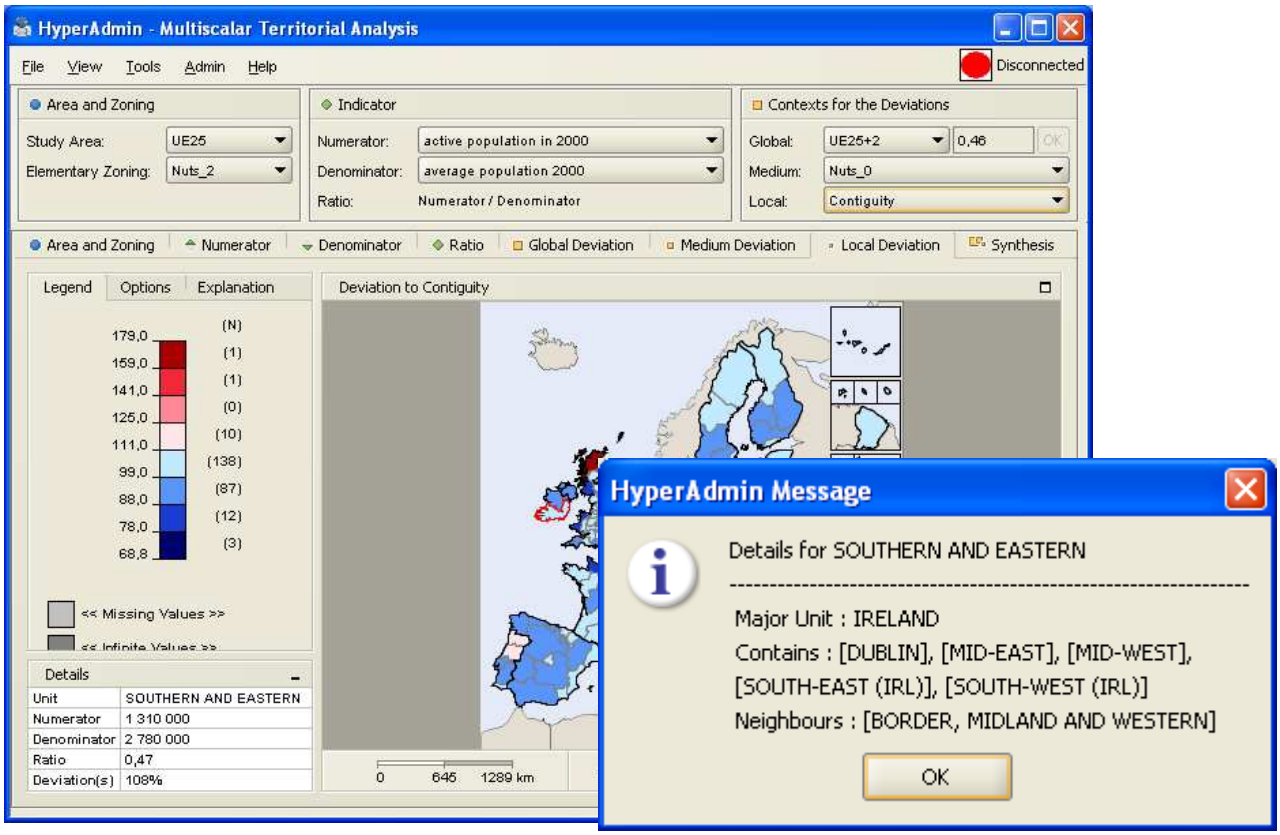


Figure 50. HyperAdmin avec seulement la Contiguïté comme relation de proximité.

Les nouveaux voisinages peuvent être choisis par l'utilisateur comme contexte de la déviation locale, (cf. figure 51).

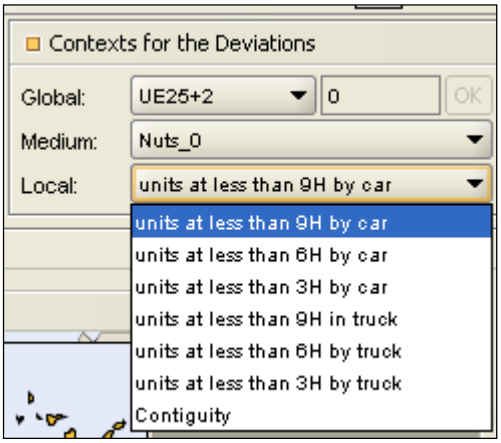


Figure 51. Choix de voisinages comme contexte pour le calcul de la déviation locale.

Lorsque qu'on utilise un voisinage fondé sur des distances voitures à moins de 3 heures, l'Irlande du sud n'a plus de voisine, comme le montre la fenêtre d'information de cette unité (cf. figure 52). Ce qui correspond aux données de la feuille Excel définissant les relations de distance temps voiture fournies en entrée.

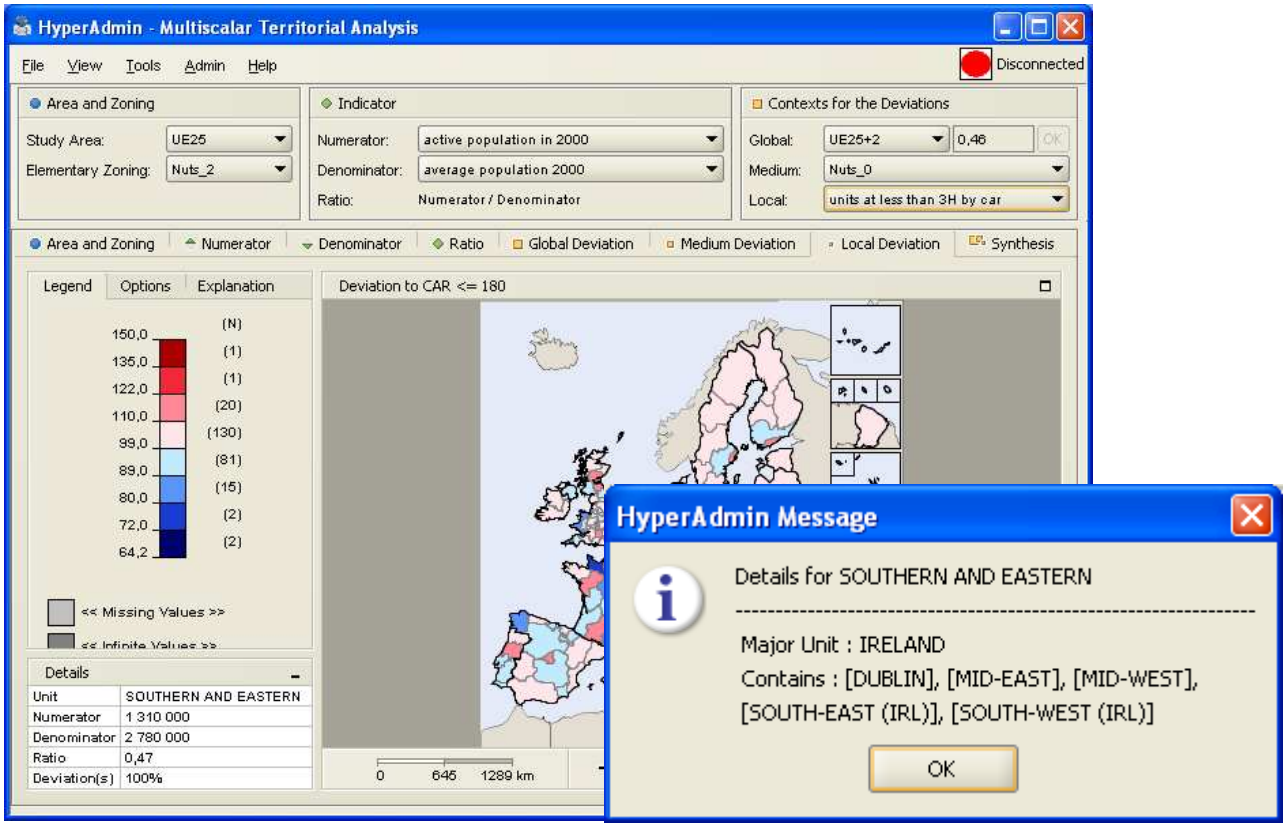


Figure 52. Carte de déviation locale fondée sur l'usage d'un voisinage voiture (distance temps inférieure à 3 heures).

b) Importer un nouveau projet en base à partir de fichiers ASCII

Le cas d'usage le plus classique est l'import en base de données d'un projet, pour pouvoir retravailler dessus ultérieurement, et surtout, pouvoir calculer les relations de contiguïté entre unités, si elles sont nombreuses. Par exemple, la Roumanie qui est le jeu de données le plus volumineux dont nous disposions, comprend 2946 communes, ce qui implique un calcul des contiguïtés via PostGIS. Sans être connecté sur une base, on peut choisir de lire des fichiers ASCII, (cf. figure 53), puis via un dialogue simple préciser l'emplacement et la nature des fichiers et le nom du nouveau projet, (cf. figure 54).

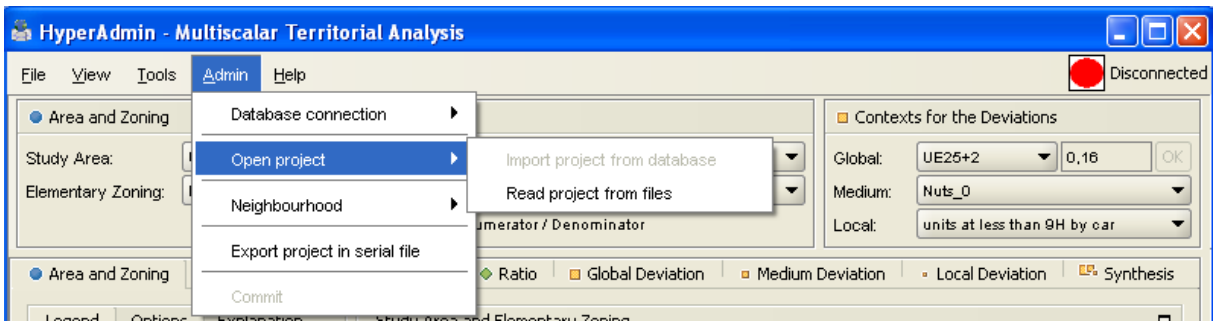


Figure 53. Menu Admin : choisir de lire un projet dans des fichiers ASCII.

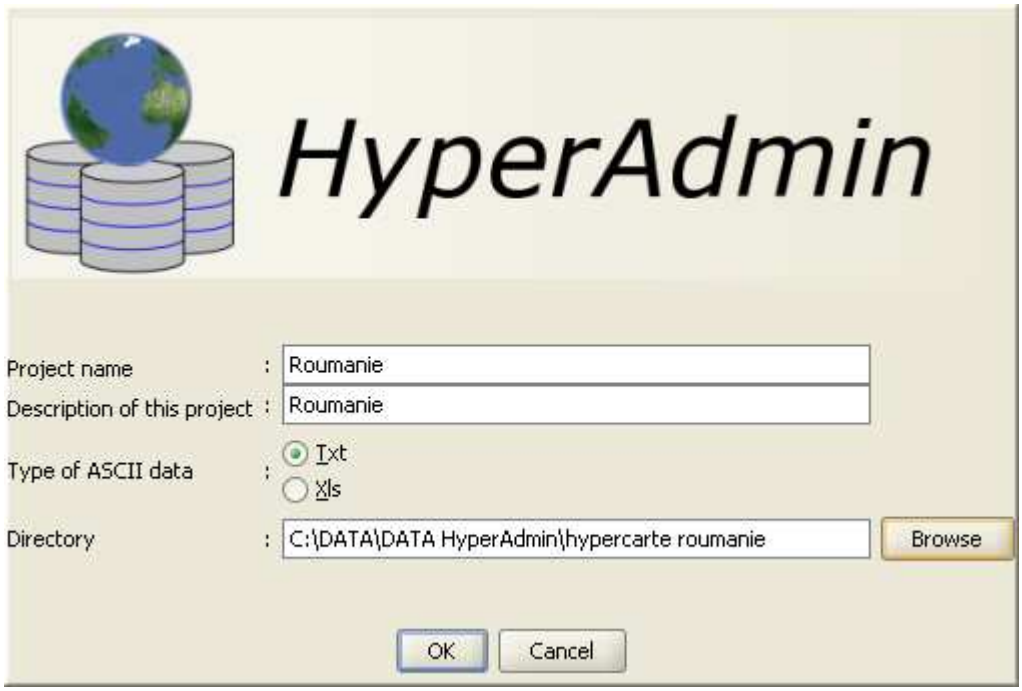


Figure 54. Donner un nom au nouveau projet, préciser l'emplacement des fichiers ASCII et leur type (Excel ou texte).

Lorsque le parcours des fichiers ASCII est terminé, un message d'information s'affiche pour expliquer que le calcul de la contiguïté nécessite de sauver le projet en base, (cf. figure 55).

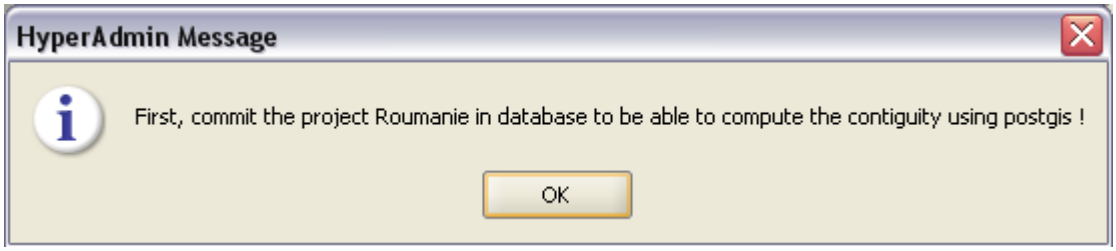


Figure 55. Message informant l'utilisateur de la nécessité d'une sauvegarde en base pour le calcul des contiguïtés.

L'interface d'*HyperAdmin* affiche alors le nouveau projet, mais aucun contexte n'est disponible pour les déviations locales comme le montre la figure suivante.

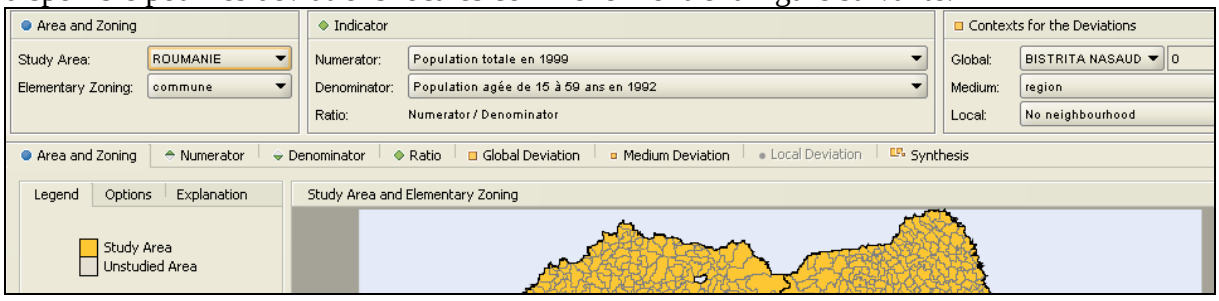


Figure 56. HyperAdmin avec le projet Roumanie, avant calcul de contiguïté.

Si l'on se connecte sur la base, on peut ensuite sauvegarder le projet via le menu « commit » (cf. figure 57). Ensuite un message indique si l'opération s'est bien déroulée (cf. figure 58).

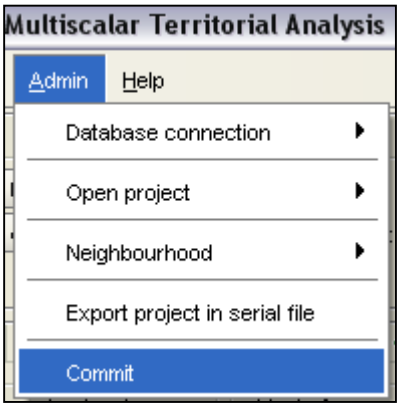


Figure 57. Menu Admin : sauvegarder en base de données.

Figure 58. Message informant l'utilisateur du bon déroulement de l'opération de sauvegarde en base.

Lorsque ces opérations ont été effectuées, il ne reste plus qu'à créer un voisinage, en demandant le calcul des contiguïtés, avec le dialogue de création de voisinages (cf. figure 49). Comme le calcul peut-être long (environ 5 minutes), une barre de défilement indique le niveau d'avancement à l'utilisateur, illustré par les figures 59 et 60.

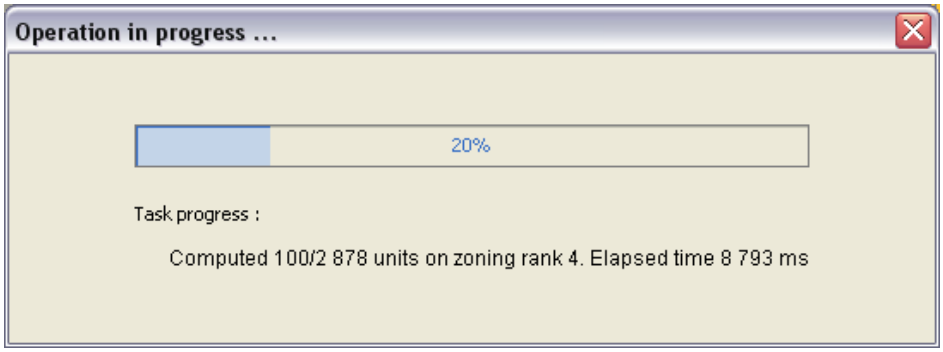


Figure 59. Barre de défilement en début d'opération.

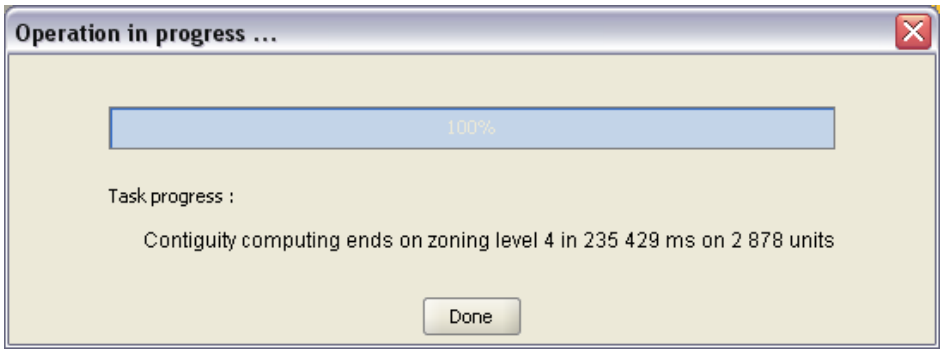


Figure 60. Barre de défilement en fin d'opération.

Ensuite, on peut exporter le projet dans un fichier de données sérialisées afin de l'exploiter dans HyperAtlas, (cf. figure 61), dont on précise le nom et l'emplacement (cf. figure 62).

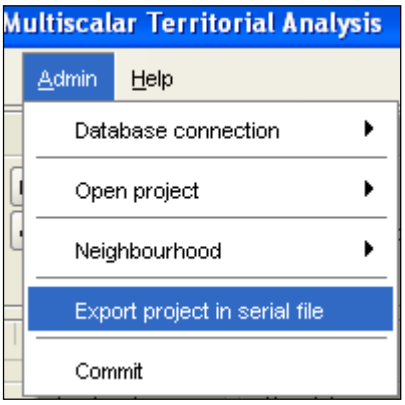


Figure 61. Menu Admin : exporter un projet dans un fichier de données.

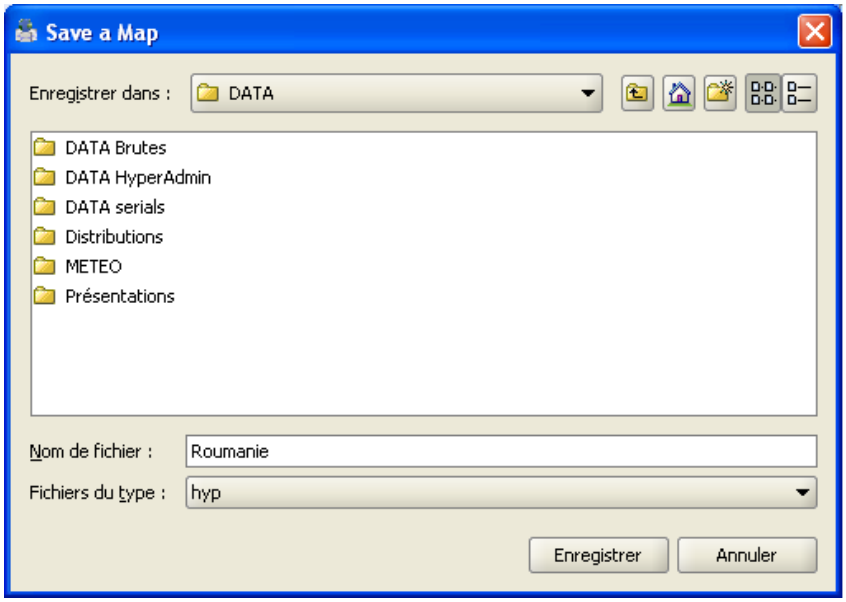


Figure 62. Choisir l'emplacement et le nom du fichier de données .hyp.

Finalement, on visualise dans HyperAtlas ce nouveau jeu de données, et on examine par exemple la répartition des personnes âgées sur le territoire roumain. On constate que les centres urbains (en violet sur la figure 63) sont plus habités par des personnes âgées que leur périphérie (les régions en vert moyen), ou que les régions les plus inaccessibles (vert foncé).

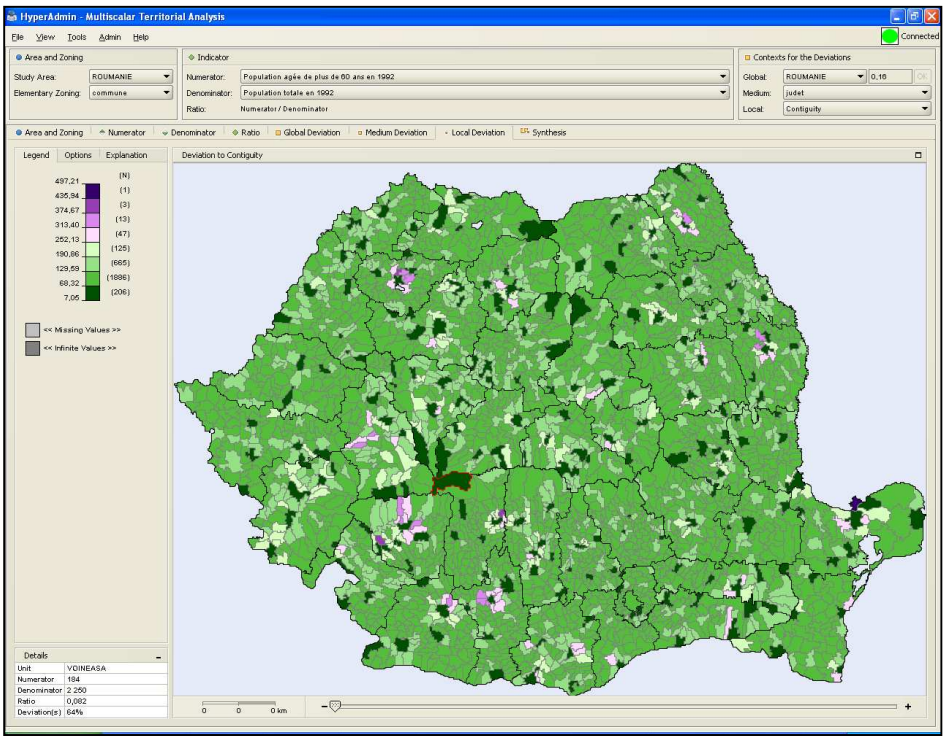


Figure 63. Carte de déviation locale du ratio "population de plus de 60 ans" sur "population totale" en Roumanie, en 1992.

4.4. Bilan

Notre action sur *HyperAdmin* se résume ainsi : extension et refonte du modèle de données pour supporter des fonctionnalités avancées, puis reprise d'un module d'acquisition de données existant pour lui apporter les améliorations nécessaires. Cette action, menée sur 5 mois, a abouti à une version livrée les trois derniers jours de janvier 2007 aux géographes parisiens. Elle a démontré avec succès que l'on pouvait alors intégrer de façon autonome le jeu de données sur la région Ile de France, par exemple. Elle permet aussi avec succès d'utiliser plusieurs types de contexte pour la déviation locale, des voisinages à 2 heures de distance voiture, 3 heures de distance camion, ou encore la contiguïté simple.

4.4.1. Performances de HyperAdmin

La nouvelle structure d'*HyperAdmin* peut-être considérée comme une base de code robuste et évolutif permettant de poursuivre les développements dans les directions indiquées par le cahier des charges. D'autre part, les performances enregistrées dans divers domaines (vitesse d'affichage des déviations locales, rapidité de fabrication des jeux de données sérialisés) sont satisfaisantes. Toutes les mesures que nous présentons ont été réalisées sur un PC de bureau (Pentium 4, 750 Mo de RAM), et en utilisant une JVM configurée avec une taille mémoire du 250 Mo, constituant en environnement très standard.

a) Performance des matrices de distance dans HyperAtlas

On mesure les temps de calcul des déviations locales mettant en jeu le calcul des relations de contiguïté. Dans la version 1.2.3, elles sont pré-calculées dans le modèle de données, alors que dans la version 1.2.2, l'ancien algorithme est encore en œuvre. Le point de mesure est la méthode *setLocalDeviationDeltas*() qui parcourt chaque unité pour connaître ses unités contiguës et calculer le ratio moyen des voisins. On vérifie que sur un maillage fin (NUTS 3), le nouveau modèle (v 1.2.3) est nettement plus performant, cf. tableau 12.

Aire d'étude	**UE25+2**		**UE15**	
	Version 1.2.2	Version 1.2.3	Version 1.2.2	Version 1.2.3
NUTS 0	15	0	16	0
NUTS 1	31	0	16	0
NUTS 2	94	16	46	16
NUTS 2-3	93	31	79	16
NUTS 3	**156**	**31**	**125**	**31**

Tableau 12. Comparaison entre ancienne et nouvelle version *d'HyperAtlas* des temps de calcul de déviation locale en ms.

b) Temps de calcul des contiguïtés

La durée nécessaire pour calculer les relations de contiguïté dans HyperAdmin est un élément important pour que l'utilisateur apprécie l'outil. Nous les avons mesurées sur plusieurs jeux de données (Rhône-Alpes, Europe, Roumanie, Cameroun, etc.). Avec ces jeux, la durée des calculs n'excède jamais les 5 minutes, et une barre de défilement apparaît pour faire patienter l'utilisateur, indiquant sur chaque niveau de maillage le nombre d'unités restant

à évaluer. Dans le tableau 13 nous précisons entre parenthèses quelle bibliothèque a été utilisée. En dessous de 1000 unités, c'est JTS qui est employée.

Europe			Rhône-Alpes		
Niveau de maillage	Nombre d'unités	Temps en ms	Niveau de maillage	Nombre d'unités	Temps en ms
1 – NUTS_0	29	1 859	1 – département	8	703
2 – NUTS_1	92	1 500	2 – arrondissement	25	860
3 – NUTS_2	280	6 547	3 – canton	311	26 468
4 – NUTS_2-3	727	52 328	4 – communes	2879	268 484 (PostGIS)
5 – NUTS_3	1329	199 0294 (JTS) 196 343 (PostGIS)			
Total	**2457**	**258 593**	**Total**	**3223**	**296 515**

Tableau 13. Mesure des temps de calcul des relations de contiguïté sur deux jeux de données.

c) Les opérations d'acquisition de données

Afin de vérifier que le logiciel HyperAdmin est suffisamment rapide pour être utilisable, nous avons vérifié les temps mis pour effectuer les différentes opérations d'acquisition de données, sur des jeux de données très différents (cf. tableau 14). Par exemple, EUROPE_ESPON inclut la définition de matrices de distance temps-camion et temps-voiture sur le NUTS 2 (280 unités) dans les fichiers ASCII, tandis qu'il n'existe pas de telles données pour les autres projets. Le projet CAMEROUN se distingue lui par sa petite taille, et le fait qu'il soit inutile de passer par PostGIS pour calculer la contiguïté.

Projets	**EUROPE_ESPON**	**CAMEROUN**	**RHONE-ALPES**	**ROUMANIE**
Volume des données ASCII	2232 Ko décrivant : 18 stocks, 5 zonings, 1329 unités de base, 8 aires, 2 distances	101 Ko décrivant : 14 stocks, 3 zonings, 214 unités de base, 69 aires	2103 Ko pour : 51 stocks, 4 zonings, 2878 unités de base, 34 aires d'étude	874 Ko décrivant : 8 stocks, 3 zonings, 2946 unités de base, 51 aires
Lecture des fichiers ASCII	42s	2s	12s	17s
Lecture d'un projet en base	35s	42s	242s	455s
Calcul des contiguïtés	166s	6s	350s	265s
Sauvegarde du projet en base	1081s	134s[34]	231s	235s
Sauvegarde de la contiguïté	360s		156s	98s
Sérialisation dans un fichier	36,3s 6106 Ko	2,8s 732 Ko	60s 8114 Ko	21,4s 7987 Ko

Tableau 14. Temps mesurés pour les diverses opérations d'HyperAdmin sur les jeux de données.

[34] Temps qui inclut la sauvegarde de la matrice de distance pour la relation de contiguïté

Excepté pour la sauvegarde des matrices de relations valuées en base (cas de Europe_ESPON), les temps de manipulation n'excèdent jamais les cinq minutes. Par ailleurs, les temps d'affichage des cartes sont identiques exactement à ceux de HyperAtlas. Et certaines opérations lentes, calcul de contiguïté, sauvegarde des contiguïtés, n'ont besoin d'être menées qu'une seule fois.

4.4.2. Perspectives d'amélioration

Par manque de temps, certaines fonctionnalités imaginées dans le cahier des charges n'ont pas été encore implémentées. Nous sommes arrivés quasiment au même niveau de fonctionnalités que celui du prototype *HyperAdmin V0.1 Alpha 2*. Ce qui laisse une très large place à de nouvelles réalisations sur ce module.

Il faudrait, en particulier, achever le menu Admin, pour offrir des possibilités de filtrage de données à l'affichage et pour l'export de données. Par ailleurs, le code pour la modification de valeurs de stocks sur des unités sélectionnées à la souris existe dans l'ancienne version : il suffit de le récupérer. En outre, la prochaine étape serait aussi de guider davantage l'utilisateur dans certaines étapes. Par exemple, s'il choisit de calculer des contiguïtés, afficher automatiquement les dialogues de connexion et sauvegarde en base.

Il nous semble aussi que le modèle de données peut encore être optimisé dans sa structure pour réduire l'espace mémoire occupé. Un analyse fine des objets est nécessaire, et peut-être que l'utilisation de matrice de distance définies comme des tableaux à double entrées dans *HyperAtlas* allègerait la taille des fichiers sérialisés.

Enfin, nous pourrions passer à une étape suivante dans le cadre d'une cartographie participative et cognitive : adapter dynamiquement l'outil (et son niveau de difficulté) aux compétences des utilisateurs. L'objectif est de cibler la catégorie de l'usager (avec par exemple une classification en profil « utilisateur », « expert », « admin » de la nomenclature établie au chapitre 3, figure 17) pour lui proposer par exemple des ratios prédéfinis si il est simple utilisateur. Ceci implique aussi de développer la gestion de méta-données sur les stocks et les utilisateurs connectés, de tracer les usages de l'interface puis de les sauvegarder pour les exploiter ultérieurement.

5. HyperSmooth : un module pour l'analyse spatiale multiscalaire

Suite au succès du module d'analyse territoriale *HyperAtlas*, la nécessité d'un module interactif permettant de s'affranchir des maillages administratifs s'est imposée. Notre objectif est de réaliser un environnement basé sur l'infrastructure du Web capable de générer dynamiquement des cartes continues. A cette fin, nous employons la méthode du potentiel. Ce choix, nous l'argumentons en expliquant les avantages de cette méthode. Nous présentons ensuite l'étude menée pour relever ce défi technique : offrir un environnement ouvert et performant, avec un haut niveau d'interactivité alors que la complexité du calcul est élevée, tout en protégeant l'accès aux données. Les réalisations sont ensuite détaillées, et nous rapportons les performances enregistrées. Nous tirons ensuite le bilan exhaustif de notre réalisation, en mettant un accent particulier sur l'ensemble des améliorations possibles.

5.1. Définition et usages de cartes de potentiel

Nous souhaitons visualiser des données fortement hétérogènes au niveau des mailles de manière à faire émerger des propriétés de la carte à un niveau supérieur (bassins d'attraction, maxima de potentiels, etc.). L'objectif est de visualiser la distribution spatiale des phénomènes analysés, à un niveau macroscopique, c'est-à-dire largement supérieur aux maillages d'observation. La méthode que nous proposons, appelée méthode de transformation par potentiel, conserve la masse totale des données et en donne une représentation non biaisée. Cette méthode, lorsque la portée du potentiel est petite, peut être appliquée à du lissage de données. Dans un premier temps, nous présentons quels sont les fondements mathématiques de cette méthode et les interprétations possibles des valeurs de potentiel. Ensuite, nous comparons le potentiel à diverses méthodes équivalentes. Enfin, nous listons des cas d'usages où ces cartes s'avèrent particulièrement utiles.

5.1.1. Principe

La méthode du potentiel permet de réaliser une cartographie continue de phénomènes spatiaux discrets. De telles représentations sont nécessaires lorsque l'on veut s'abstraire du maillage territorial (parce que ce maillage est hétérogène, ou parce qu'il n'est pas signifiant pour le phénomène étudié) pour ne garder que l'organisation spatiale du phénomène, sans référence au découpage sous-jacent du territoire. Ce type de méthode est aussi une solution à la représentation de phénomènes mesurés à des niveaux de maillage différents. Du point de vue méthodologique, elle s'apparente aux méthodes de traitement du signal par déconvolution du signal échantillonné.

a) Fondements mathématiques

Nous désirons projeter un espace géographique donné, sur lequel est plaqué un maillage constitué d'unités territoriales et de leur stock (ou indicateur statistique) associé, sur un espace géométrique 2D. L'indicateur échantillonné, environnemental ou social ou économique, résulte d'un comptage des effectifs sur la maille : c'est un entier positif. On rattache à cet échantillon un attribut spatial, qui est en général le centroïde ou bien le centre de gravité de la maille. On observe que les stocks sont additifs et ont un effet proportionnel à leur valeur. Pour cela, nous discrétisons l'espace géométrique, et en chacun de ses points nous évaluons le potentiel chaque stock.

Soit *A* l'ensemble des unités territoriales, *a* un élément de cet ensemble, *Sa* la valeur du stock sur cette unité. Sachant que les effets des stocks s'additionnent et sont liés à la distance δ entre *a* et le point *M*, nous définissons le potentiel *Φ(M)* en tout point *M* de l'espace géométrique par : $\Phi(M) = \sum_{a \in A} S_a f(\delta(a, M))$

[1]

Par exemple, si *A* est l'ensemble des communes européennes (ou équivalents dans le NUTS 5), *a* la commune de Nuoro en Sardaigne, alors *Sa* serait le nombre de personnes centenaires vivant à Nuoro.

On spécifie la distance *δ(a,M)* en mesurant l'éloignement *d* entre *M* et $g_{a,,}$ un point représentatif de *a* (qui peut être son centre de géométrie, son centre administratif ou industriel, etc.). La contribution au potentiel de chaque élément *a* est pondérée par une fonction *f* de la distance *d*, car l'effet d'un stock diminue usuellement avec la distance, et il est maximal à une distance nulle, et nul à une distance infinie.

Il est important de noter que les fonctions *f* que nous proposons sont normalisées, autorisant ainsi à donner une interprétation probabiliste du potentiel :

$$\int_{R^2} f(d(g_a, M)).dM = 1 \qquad [2]$$

Nous contrôlons que la somme totale des stocks égale celle des potentiels calculés (l'invariant de masse) qu'exprime l'équation [3], et on obtient ainsi une redistribution de la masse sur l'espace considéré. :

$$\int_A \Phi(M).dM = \sum_{a \in A} S_a \qquad [3]$$

De l'équation [1], nous déduisons que la complexité du calcul dépend à la fois de la taille de l'espace administratif (le nombre *n* d'éléments *a*), et de la résolution de l'image à produire (le nombre *m* de points *M* que l'on estime).

b) Interprétation du potentiel

Nous proposons une interprétation du potentiel dans le cadre d'un modèle de mobilité. Dans une métaphore du modèle de gravité, (cf. figure 65), *Φ(M)* s'interprète comme l'attraction exercée par l'environnement sur un mobile placé en *M*, dont le vecteur de déplacement serait alors *–grad Φ*. Une interprétation duale serait aussi que *Φ(g_a)* mesure l'influence d'une masse placée en g_a sur l'ensemble des points M de son voisinage (cf. figure 64) en considérant que chaque masse correspond en fait à la valeur d'un stock, concentrée sur le point représentatif de l'unité.

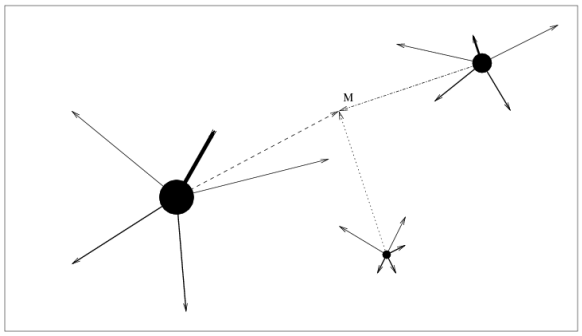


Figure 64. Interprétation stochastique.

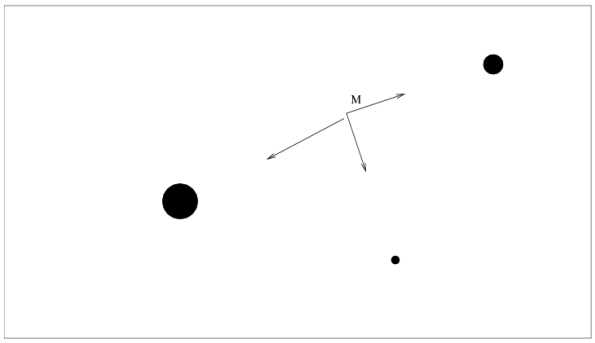


Figure 65. Interprétation gravitationnelle.

Grâce à la normalisation [2] nous pouvons aussi interpréter Φ(M) comme étant la valeur probable que nous pourrions mesurer en ce point de l'espace, pourvu que le phénomène suive la loi de dispersion qu'exprime la fonction f, et ceci si l'expansion du phénomène se passe de façon homogène et isotrope.

L'analyse dépend en effet du type de **fonction** d'interaction choisie. On dégage trois catégories : les fonctions discrètes, les exponentielles négatives, et enfin les puissances inverses. Ce choix n'est pas anodin : il dépend de l'hypothèse émise sur le mode de propagation du phénomène étudié. Par exemple, les épidémies touchant l'homme, ou bien les pièces d'euros vont se diffuser en suivant le modèle de mobilité des humains, et leur diffusion se fera soit sur de longues distances, soit sur de courtes distances, suivant le rayon d'action de l'élément contaminant [Grasland *et al.*, 2005a]. C'est pour cette raison qu'il est important que l'utilisateur puisse ultérieurement tester et valider ses hypothèses, en ayant le choix de la fonction d'interaction qu'il souhaite appliquer.
Nous résumons ici les propriétés de quelques fonctions représentatives de ces différents groupes. Leur formule est ajustée pour remplir le critère [2].

- Circulaire de rayon R

Elle correspond à un concept de voisinage spatial discret de forme circulaire. Malgré sa simplicité, elle n'est pas souvent utilisée parce qu'elle introduit des discontinuités dans la représentation des phénomènes, les cartes résultantes ressemblant à des cartes à disques, avec l'apparition d'aires circulaires de haute densité. De plus, elle ne correspond pas à la perception d'un espace continu que l'homme possède. On lui préfère alors souvent sa variante, la fonction à disque amorti.

- Exponentielle de paramètre α

En général, elles décrivent bien les phénomènes de migration à courte distance.

- Gaussienne de paramètre β

La fonction Gaussienne est un cas particulier de la fonction exponentielle. Elle correspond aux théories développées par Hägerstrand sur les champs d'information et par Stouffer sur les opportunités intermédiaires, et c'est pourquoi elle est préféré par Claude Grasland pour étudier les potentiels migratoires. Le potentiel qu'elle calcule est continu et dérivable en tout point, ce qui donne l'avantage de pouvoir établir des cartes de discontinuité,

où l'on est certain que les ruptures observées sont causées par le phénomène étudié. De même, sa dérivabilité autorise la fabrication de cartes de gradient pour visualiser des lignes de répulsion/attraction du phénomène observé

- Pareto de paramètre γ

Elle décrit mieux les probabilités de migration à longue portée qu'une exponentielle négative, cependant l'impact lointain sera particulièrement affaibli.

En résumé, et pour donner une représentation visuelle de l'influence de ces fonctions sur la forme de la surface de potentiels calculés, voici quelques représentations 3D (cf. figures 66 à 69) établies avec Maple. A partir de données fictives (cf. tableau 15), les potentiels z ont été calculés en tout point (x,y) d'une grille de taille 10×10 en variant le type de fonction d'interaction. Les points G_1, G_2, G_3 correspondent aux centroïdes de trois unités sur lesquelles on a compté un stock (la valeur du stock correspond à leur masse).

Points	Coordonnées	Masse
G_1	(2, 6)	2
G_2	(5,5)	4
G_3	(6, 1)	1

Tableau 15. Exemple d'échantillonnage pour le potentiel

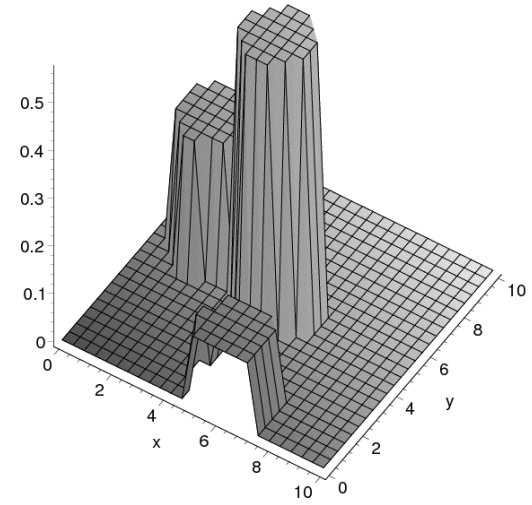


Figure 66. Lissage par fonction circulaire.

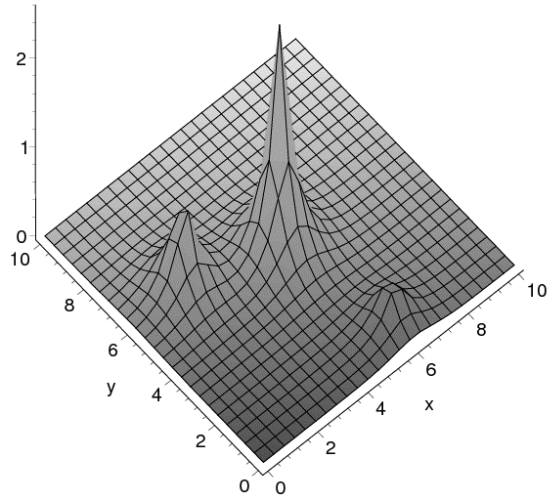


Figure 67. Lissage par fonction gaussienne.

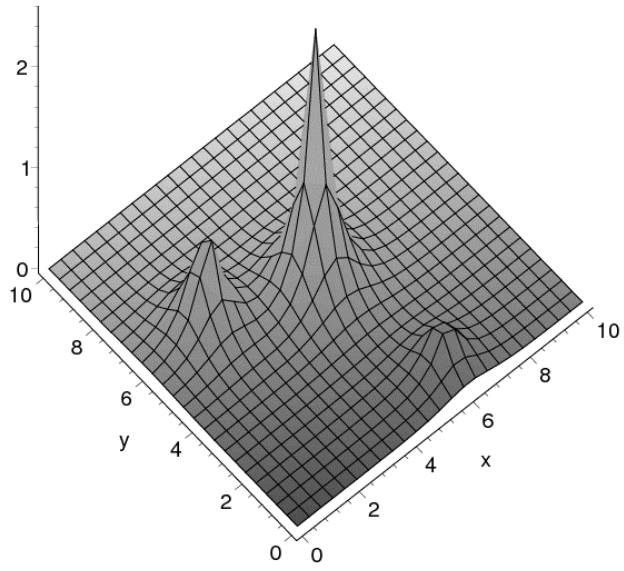


Figure 68. Lissage par fonction exponentielle.

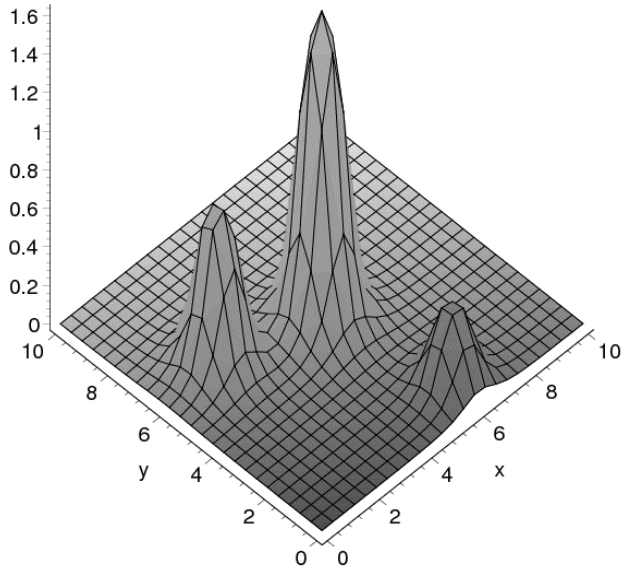


Figure 69. Lissage par fonction de Pareto.

On remarque que les masses possèdent un impact d'un niveau plus faible sur leur proche voisinage avec l'emploi d'une fonction inverse modifiée comme Pareto (cf. figure 69), que dans le cas d'une fonction exponentielle (cf. figure 68), mais cet impact porte plus loin. Et la discontinuité des cartes produites avec la fonction circulaire apparaît nettement sur la surface de la figure 66.

c) Paramétrage de l'analyse.

Le choix de « bons » paramètres pour l'analyse est délicat. Comme nous l'avons déjà exposé, le choix de la fonction dépend de l'hypothèse émise sur le mode de propagation du phénomène analysé.

Après avoir discuté le type de fonctions disponibles pour calculer un potentiel, il nous reste à faire observer que ces fonctions ont un paramètre, R, α, β, ou γ, qui influence leur forme et donc leur rayon d'action. Nous définissons alors la **portée** p (ou voisinage) comme la distance moyenne d'action d'une masse (placée en un point O par exemple) sur son voisinage. La portée est reliée à la forme de la fonction d'interaction par l'équation suivante :

$$p = \int_{R^2} d(O,M)\, f(d(O,M)).dM = \int_0^{+\infty} f(r) 2\pi r^2 .dr \qquad [4]$$

La portée peut être interprétée comme l'échelle spatiale de représentation choisie. Le tandem (fonction, portée) traduit les hypothèses économiques et sociologiques associées aux interactions entre les acteurs sur le territoire. Pour les différentes fonctions introduites, le tableau 16 résume le lien existant entre paramètre de fonction et portée moyenne.

Modèle	Fonction	Paramètre	Portée
Disque	$f(d)=\frac{1}{\pi R^2}d \quad si \quad d(g_e,M)<=R$ $f(d)=0 \quad si \quad d(g_e,M)>R$	**R**	$\frac{2}{3}R$
Disque amorti	$f(d)=\frac{3}{\pi R^2}\left(1-\left(\frac{d}{R}\right)^2\right)^2 \quad si \quad d(g_e,M)<=R$ $f(d)=0 \quad si \quad d(g_e,M)>R$	**R**	$\frac{16}{35}R$
Exponentiel	$f(d)=\frac{\alpha^2}{2\pi}e^{-\alpha d}$	**α**	$\frac{2}{\alpha}$
Gaussien	$f(d)=\frac{\beta}{\pi}e^{-\beta d^2}$	**β**	$\frac{1}{2}\sqrt{\frac{\pi}{\beta}}$
Pareto	$f(d)=\frac{2\gamma}{\pi^2(1+\gamma^2 d^4)}$	**γ**	$\sqrt{\frac{2}{\gamma}}$

Tableau 16. Relation entre portée et paramètre de forme. Source : [Grasland et al. 2006].

A portée identique, nous pouvons calculer le ratio de deux potentiels de stocks différents, ratio qui s'interprète facilement comme une densité, puis comparer des densités sur des portées différentes. Par exemple, sur une portée R_1 donnée fixe et uniforme, le potentiel *N* de richesse peut se rapporter au potentiel *D* du nombre d'habitants, ce qui nous donne une densité de richesse $Z(R_1)$ (le PNB par habitant sur une portée R_1). Il est alors intéressant de comparer par exemple ce taux de richesse Z centré sur un point de la carte, par exemple Saint Dié dans les Vosges, en utilisant les portées successives de 250 km, 1000 km, puis 5000 km pour s'apercevoir qu'il diminue lorsque l'on s'éloigne de ce pôle de prospérité [Grasland, 2003]. La carte de la figure 70 est basée sur la différence logarithmique des PNB par habitant dans deux voisinages gaussiens de portée différente (250 et 1000 km) : log ($Z(R_1)$-$Z(R_2)$). Il est donc pertinent que l'outil donne le choix libre de la portée à l'utilisateur.

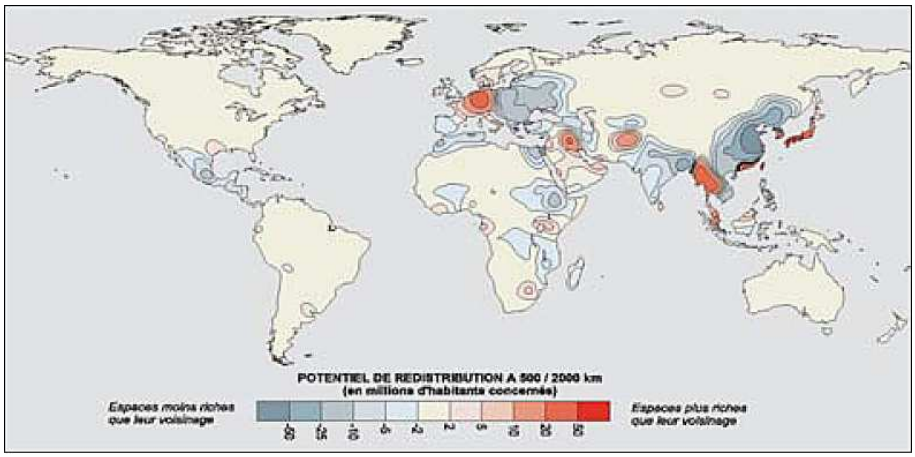


Figure 70. Potentiel de redistribution des richesses et populations sur des distances de 500 à 2000 km. Source : Grasland, 2003, Mappemonde n°69

La portée ne peut pas être trop réduite sous peine d'introduire des erreurs. Sa valeur minimale dépend de la finesse de l'échantillonnage des données. Le lien entre la taille de la maille et la portée du filtre est donné par l'analogue du théorème de Nyquist [Nyquist, 1928]. La fréquence de l'échantillonnage d'un signal doit être supérieure à deux fois la fréquence du signal pour garantir la reconstruction du signal à partir de la transformée. Dans notre cas, cela signifie que la méthode devient imprécise si la portée est plus petite que deux fois la taille maximale des mailles. Et empiriquement, Claude Grasland vérifie que en moyenne, un lissage sur l'espace européen échantillonné au niveau du NUTS 3 (niveau départemental) est optimisé avec une portée de 80 km car sa corrélation avec un lissage à partir de données de niveau NUTS 2 est alors maximale, [Grasland et al., 2006].

Le type de **distance** que l'on utilise dépend lui de deux facteurs. D'une part, de la taille de l'espace couvert par la carte, car à l'échelle d'un continent par exemple, on ne peut pas utiliser la distance euclidienne sans introduire des déformations (on observerait un rapprochement excessif des points en bordure de la carte). La distance orthodromique convient alors mieux car elle tient compte de la sphéricité de la Terre. D'autre part, du type de phénomène que l'on analyse : en effet il faut aussi parfois pouvoir tenir compte du relief ou de l'accessibilité relative des lieux. Il est par exemple plus rapide de se rendre de Lyon à Paris en TGV plutôt que de Grenoble à Lyon, alors qu'en distance euclidienne ou orthodromique, Grenoble est plus proche de Lyon que Paris. C'est pourquoi, par la suite, on voudrait utiliser d'autres topologies, rendant compte de l'anisotropie de l'espace (distance temps voiture par exemple). Mais notre proposition actuelle se limite à la distance orthodromique.

5.1.2. Exploration des méthodes locales

Nous situons ici les avantages et les inconvénients de notre méthode, et justifions notre démarche, au regard de ce qui existe.

Pour des données correspondant à des comptages administratifs, Claude Grasland démontre que les méthodes locales fondées sur l'interpolation spatiale (telles que celles décrites en annexe du document) sont inadaptées car elles ne font pas disparaître le maillage sous-jacent, et au contraire, elles pourraient faire croire que les disparités observées sont le fait du phénomène observé, alors qu'en vérité elles sont la résultante de l'hétérogénéité du maillage [Grasland et al, 2006]. Sa démonstration s'appuie sur la comparaison de cartes obtenues à partir d'un maillage de niveau NUTS 2 ou NUTS 3 en employant la méthode de Shepard. Or, ce sont des méthodes de ce type qui sont implantées dans la majorité des SIG (Système d'Information Géographique). Cette inadéquation vient du fait qu'elles sont conçues pour fonctionner sur un semis de points correspondant à des mesures locales (comme on mesure une température en un lieu repéré par des coordonnées GPS, ou bien on relève une quantité de minerais sur des forages précis). Mais notre méthode doit fonctionner sur des unités aires, et non des lieux précis, sur lesquelles on a effectué un comptage d'un indicateur dont on ignore la répartition à l'intérieur de chaque unité.

Pour cette raison, la méthode la plus proche est la méthode pycnophylactique [Tobler, 1979], qui elle aussi conserve la masse des données. En effet, elle prend en compte le mode d'échantillonnage des données, et elle réalloue les stocks mesurés à l'intérieur de chaque maille de façon à :

1. garantir la continuité avec une autre maille voisine
2. conserver sur chaque maille la masse mesurée.

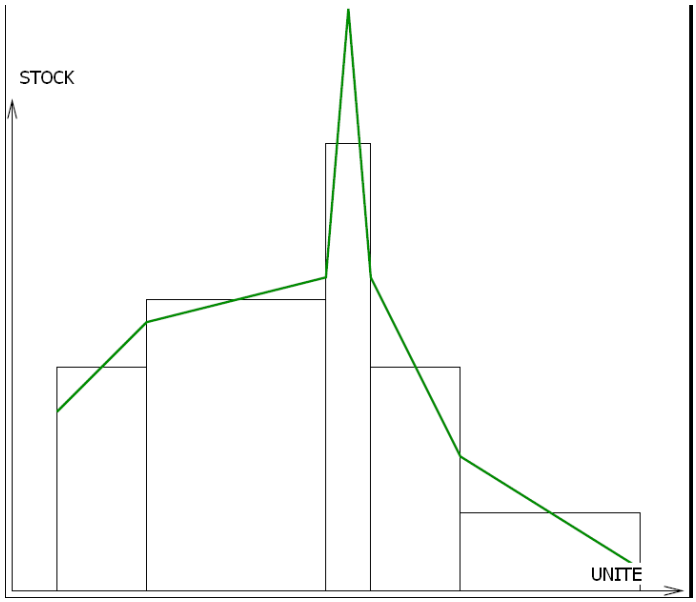


Figure 71. Réajustement de la répartition des stocks sur des unités d'aires voisines via la méthode pycnophylactique.

La courbe continue verte de la figure 71 montre comment s'effectue le réajustement de la répartition des stocks à l'intérieur de chaque unité. C'est un cas simple, on peut employer des méthodes d'ajustement plus complexes où les surfaces obtenues sont au moins dérivables, voire de classe supérieure. Cependant, en raison du second postulat, la visualisation dépend alors du maillage utilisé pour l'analyse : les résultats seront différents selon le niveau d'agrégation des données, alors que ce n'est pas le cas avec le potentiel.

Il faut noter aussi que notre méthode présente des limitations qui sont la conséquence du mode de collecte des données. L'inconvénient majeur vient de ce que les données sont échantillonnées sur un espace limité, sur les bordures duquel la méthode ne peut fonctionner de façon isotrope : les points situés en dehors de cet espace qui pourraient avoir un poids et une influence non négligeables ne sont pas pris en compte. L'étendue de l'aire présentant un défaut d'évaluation est directement proportionnelle à la portée de l'analyse. Par exemple, Laure Charleux explique que lorsque qu'on lisse le Produit Intérieur Brut (PIB) de l'espace constitué de l'Allemagne et le Bénélux avec un rayon de lissage de 100 km, on se rend compte que la Saxe, zone frontalière d'avec la Tchéquie, semble plus pauvre qu'elle ne l'est, alors que la présence proche de l'îlot de prospérité praguois devrait augmenter son potentiel de richesse [Charleux, 2003]. La figure 72 situe la Saxe, et Prague, et la zone de biais. On remarque sur la carte choroplèthe de l'Europe des 25 que Prague est une région plus riche que ses voisines (en rose), puisque nous visualisons la déviation locale du rapport PNB sur population en 1999.

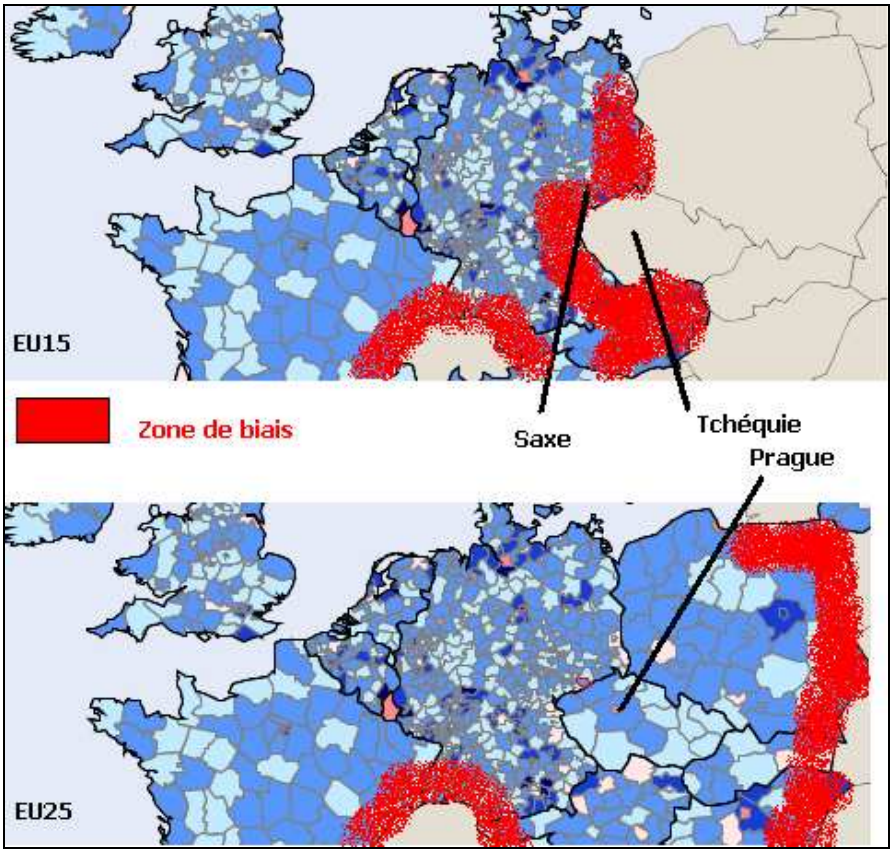


Figure 72. Figuration des zones de biais induites sur les frontières d'une aire d'étude. Les cartes choroplèthes de l'Europe de 15 ou des 25 montrent la déviation locale du rapport du PNB à la population en 1999.

De même, l'hétérogénéité du maillage administratif gène l'analyse. En effet, la portée minimale en dessous de laquelle la méthode devient imprécise nous est donnée par l'analogue du théorème de Nyquist [Nyquist, 1928] : elle vaut deux fois la taille maximale des mailles. Mais sur les vastes aires scandinaves du NUTS 3, la portée minimum est doublée par rapport à celle que l'on pourrait utiliser sur la région allemande et Bénélux. Si on prend un rayon adapté pour les régions scandinaves, alors on généralise presque trop l'information sur les régions du cœur de l'Europe. La figure 73 illustre la différence de taille des régions administratives sur le NUTS 3, correspondant aux départements en France.

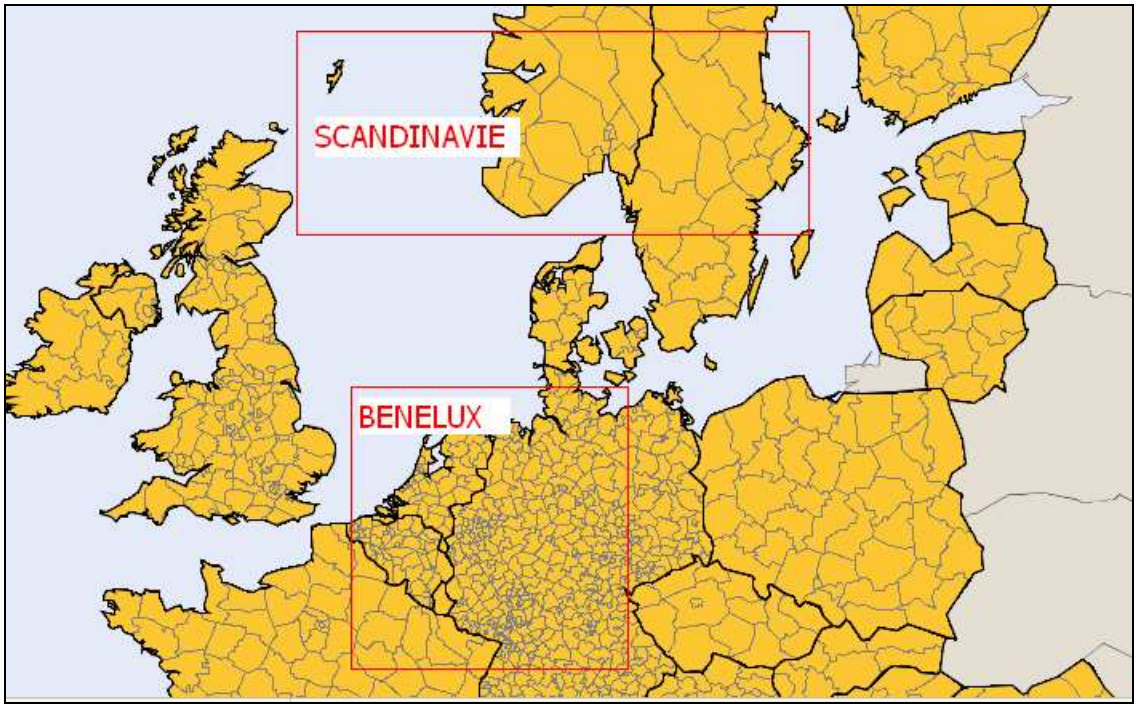


Figure 73. Illustration de l'hétérogénéité du maillage de niveau NUTS_3 sur l'Europe.

Malgré les inconvénients cités, nous illustrons des cas d'utilisation dans le paragraphe suivant où cette méthode révèle sa force.

5.1.3. Usages particuliers

Outre la restitution de la continuité de phénomènes sociaux que nous avons déjà présentée dans le cadre général, la méthode révèle sa force dans les cas suivants :

- Etudier les évènements rares

En réglant une portée optimale (celle qui permet de conserver le maximum de différenciations spatiales tout en évitant d'interpréter des maxima non significatifs). Un bon exemple est fourni par la contribution de C. Grasland à l'étude de la « zone bleue » en Sardaigne où la visualisation des potentiels sur une portée courte (15 km) met en évidence une zone de présence étonnamment forte de centenaires [Poulain et al, 2004], illustrée par la figure 74.

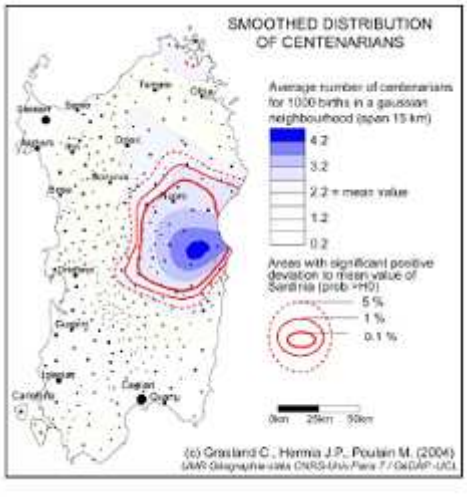
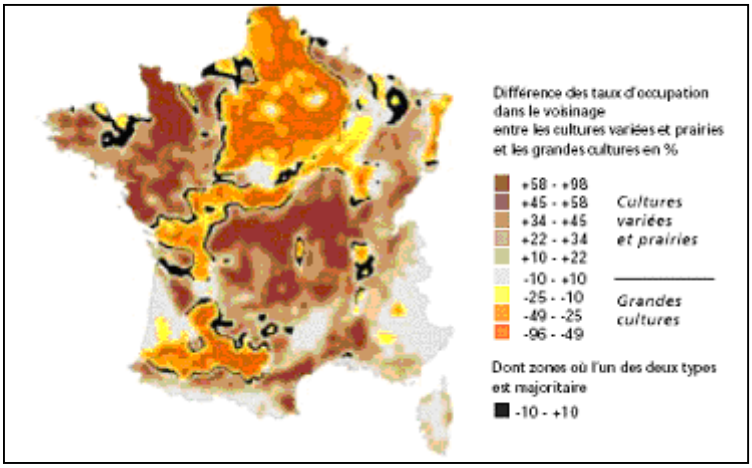


Figure 74. Cartographie des différences significatives entre quantité de centenaires observés et distribution théorique, réalisée par un potentiel gaussien de portée 15 km. Source : Poulain, Mario Pes, Grasland & al., (2004)

- Filtrer des distributions complexes



Nous avons choisi comme exemple celui de la répartition de l'occupation des sols sur le territoire français. Une étude réalisée en 1998-1999 dégage ainsi des continuités par grands types d'occupation des terres et visualise les compétitions entre des usages divers (cf. figure 75). La méthode a été ensuite transposée à l'échelon européen dans le cadre du projet CORILIS. [Lacaze et al, 2000]

Figure 75. Visualisation de divers usages du sol en France, basée sur un potentiel gaussien de portée 20 km. Source : Lacaze & Nirascou, 2000.

- Assurer la confidentialité des données observées

Employée pour étudier les inégalités scolaires dans l'agglomération parisienne à l'aide de données originales sur les collèges, la méthode montre comme autre atout qu'elle préserve la confidentialité des données (en localisant sans stigmatiser un collège particulier). [François J-C, 1996], cf. figure 76.

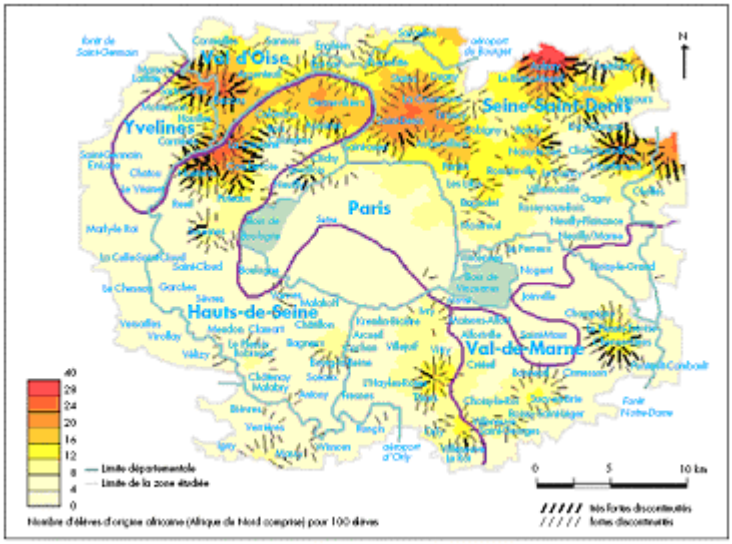


Figure 76. Cartographie des concentrations d'élèves originaires du Maghreb en 1989 en Ile de France, basée sur un potentiel gaussien de portée 5 km. Source: François J-C, 1996.

Cette étude est l'occasion de souligner que l'accès aux données de notre application doit être protégé et sécurisé. En effet, elles sont la plupart du temps confidentielles soit à cause de leur caractère payant, ou parce qu'elles sont sensibles (comme le recensement du nombre d'élèves d'origine nord-africaine dans les établissements scolaires franciliens).

Nous constatons que cette méthode est intéressante pour les géographes puisqu'elle leur permet de révéler la structure spatiale de bien des phénomènes, quelle que soit leur nature : économique, sociale, environnementale, épidémiologique. Il s'agit ensuite de concevoir une architecture adaptée aux besoins de nos utilisateurs.

5.2. Etude technique

Le volume de données à traiter étant important et les ressources exigées pour les calculs aussi, nous avons retenu l'idée suivante : déporter la partie graphique avec la visualisation du côté d'un client Web léger, tandis que les calculs et le traitement lourd de données seront effectués sur un serveur accessible à distance, via un protocole qui reste à déterminer. Le traitement côté serveur des données sera optimisé de façon à produire des résultats intermédiaires dans des temps n'excédant pas quelques secondes.

5.2.1. Choix du protocole réseau

Le protocole d'échanges entre le client et le serveur doit :

- permettre l'activation du serveur à distance sur des requêtes spécifiques,
- la récupération rapide des résultats,
- assurer la confidentialité des données, et l'authentification des utilisateurs.

En effet, à partir de trois cartes récupérées depuis le serveur, il est possible de reconstituer par interpolation les données statistiques originelles. Nous nous sommes intéressés aux protocoles de type RPC (*Remote Procedure Call*) permettant l'invocation de méthodes à distance : Java RMI, corba, SOAP ou XML-RPC.
Corba a été écarté à cause de son manque de compatibilité avec les politiques de sécurité autour des pare-feux, car les ports de communication spécifiques de Corba sont généralement fermés.
Java RMI offre de très bonnes performances réseaux et une bonne ergonomie pour le programmeur, mais induit deux risques :

- manque de stabilité du serveur si la JVM (*Java Virtual Machine*) ne supporte pas la montée en charge ou les multiples appels récursifs, à cause d'une surcharge de la pile d'exécution,
- difficulté de configuration du réseau à cause de la nature des solutions proposées par SUN pour passer à travers les firewalls.

Le dernier choix serait d'offrir un service web, accessible soit via le protocole SOAP (*Simple Object Access Protocol*) ou via XML-RPC. Ce type de technologie n'induit aucun des deux risques mentionnés précédemment. Par contre, ces protocoles étant extrêmement verbeux, il faudra faire attention à diminuer l'occupation de la bande passante. Pour cela, on pourra envisager soit de compresser les données (avec *gzip* par exemple) soit de diminuer la précision des données transférées. Nous sommes aussi conscient que ce choix de technologie « à la mode » est risqué du fait du manque de stabilité de la technologie (jeunesse du protocole SOAP, et XML-RPC déjà dépassé).

Pour assurer la sécurité et le cryptage des communications, nous avons évalué deux mécanismes, combinés avec l'une des technologies RMI citées : SSL (*Secure Socket Layer*) ou bien SSH (*Secure SHell*). SSH a l'avantage d'être un peu moins gourmand en ressources que SSL et de proposer *de facto* l'authentification, mais son usage force le client à installer une bibliothèque spécifique, ce qui complexifie l'installation du client, et pourrait à terme

nuire à sa diffusion, puisque nous visons un public d'utilisateurs non informaticiens, et non avertis des politiques de sécurité des réseaux. De plus, nous n'avons aucune garantie que le port 22 ne sera pas filtré chez le client (si par exemple, il est installé derrière un pare-feu d'entreprise qui filtre ce port). Au contraire, SSL n'oblige pas le client à installer des bibliothèques particulières. C'est pourquoi nous le préférons même s'il est un peu moins performant que SSH.

5.2.2. Stratégie d'optimisation

Remarquons, en premier lieu, que les stratégies classiques de gestion de cache ne sont pas adaptées à notre cas car chaque requête doit générer un résultat global qui ne peut être pré-calculé puisqu'il dépend des paramètres de l'analyse. Cependant, un examen approfondi des tâches de calcul sur le serveur montre qu'il existe des redondances que nous pourrions exploiter pour optimiser certaines parties du calcul. En effet, lorsque nous désirons calculer en tout point M d'une grille le potentiel $\Phi(M)$ (confère l'équation [1]), deux problèmes se posent :

1. La somme se fait sur un nombre important d'éléments
2. Le calcul des distances orthodromiques[35] $d(M, g_e)$ est coûteux puisqu'il fait intervenir des calculs d'angles en arccosinus, cosinus et sinus.

Pour optimiser le calcul, on peut jouer sur les deux facteurs. D'une part, en réduisant le nombre d'éléments e sur lesquels s'effectue la somme, et d'autre part, en tabulant de façon fine les distances.

Nous réduisons la somme en pratiquant une politique d'élagage (*cut-off)* algébrique. Les géographes utilisent habituellement un *cut-off* qui limite le calcul sur un certain rayon de portée. Cette méthode repose sur l'hypothèse qu'en dehors d'une aire définie par le cercle de rayon r (choisi arbitrairement), et de centre M, les données sont trop lointaines en termes de distance pour impacter l'estimation du potentiel en M. Notre algorithme de *cut-off* utilise une méthode numérique par recherche dans un arbre quaternaire (*quadtree*) et avec calcul récursif du potentiel à partir de la racine. Cette méthode à l'avantage de tenir compte de points éloignés mais dont le poids S_e (valeur statistique) influence le résultat du calcul.

A chaque feuille de l'arbre de profondeur n est associé un point M valorisé avec ses coordonnées (x,y) et son stock. Chaque point est ensuite sommé par groupe de 4 (des voisins sur la grille). On regarde la valeur du stock sur le niveau n-1 : si elle est valable on utilise les points « cousins » pour réévaluer le stock. Sinon, on élague cette branche. La figure 77 donne un exemple de construction de l'arbre à partir d'un espace discrétisé contenant quelques échantillons A,B, …, K.

[35] Rappel : la distance orthodromique entre 2 points A et B de coordonnées respectives A(a1, a2) et B(b1, b2) avec r rayon terrestre et les latitudes, longitudes en radians vaut : d(A,B) = arccos(sin(latA) sin(latB) + cos(latA) cos(latB) cos(longB-longA)) * r.

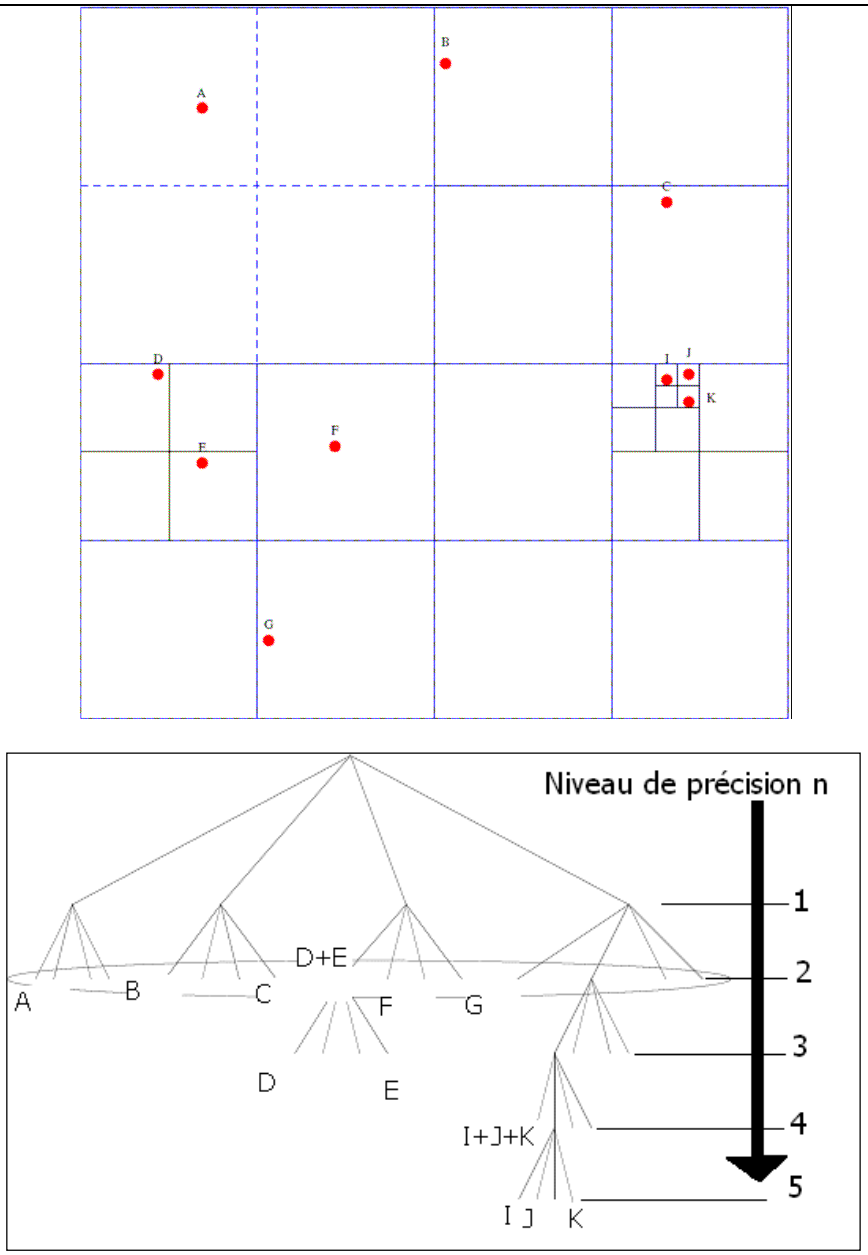


Figure 77. Exemple de construction de l'arbre à partir d'un espace d'étude discrétisé.

Le test de la quantité négligeable se fait ainsi :

$$(\sum_{noeud_{n-1}} S_e) * d_{\min} \leq \varepsilon * \Phi(M)_{cumulé} \qquad [5]$$

Si la relation [5] est vérifiée, alors les enfants du nœud $_{n-1}$ sont ignorés pour le calcul du potentiel (leur poids est négligeable). Le point délicat est d'ajuster la valeur de l'epsilon

correctement de manière à ne pas négliger trop de points. On peut prendre un millième de la somme totale des stocks par exemple.

En outre, l'évaluation d'expressions contenant des termes en arccosinus, cosinus et sinus d'angles est un autre facteur ralentissant le calcul, et il peut être contourné en tabulant de façon fine ces fonctions. C'est-à-dire que nous pré-calculons les valeurs des fonctions sur des angles correspondant à une division régulière et fine de [-π/2, π/2], (cf. tableau 17), et pour tout angle nous approximons la valeur de la fonction par la borne inférieure de la division à laquelle il appartient.

Division de [-π/2, π/2], en *n* parts : *i* varie de 0 à n	$-\frac{\pi}{2}$	$-\frac{\pi}{2}+\frac{\pi}{n}$	$-\frac{\pi}{2}+2\frac{\pi}{n}$		$-\frac{\pi}{2}+i\frac{\pi}{n}$		$\frac{\pi}{2}$
Valeurs précalculées de *f* (arccosinus, sinus ou cosinus)	$f\left(-\frac{\pi}{2}\right)$	$f\left(-\frac{\pi}{2}+\frac{\pi}{n}\right)$	$f\left(-\frac{\pi}{2}+2\frac{\pi}{n}\right)$	• • •	$f\left(-\frac{\pi}{2}+i\frac{\pi}{n}\right)$	• • •	$f\left(\frac{\pi}{2}\right)$

Tableau 17. Tabulation de fonctions.

f(x) est alors calculée de la manière suivante :

$$\left(-\frac{\pi}{2}+i\frac{\pi}{n}\right)\le x<\left(-\frac{\pi}{2}+(i+1)\frac{\pi}{n}\right) \quad \Rightarrow \quad f(x)\approx f\left(-\frac{\pi}{2}+i\frac{\pi}{n}\right) \qquad [6]$$

Par ailleurs, nous optimisons encore la plage de valeurs tabulées en la réduisant aux valeurs utiles. En effet, notre zone de traitement possède une certaine amplitude en longitude, et en latitude. En dehors de ces deux fourchettes, nous n'aurons aucun angle à calculer. Donc nous réduisons la plage [-π/2, π/2] à [*LongMin, LongMax]* et *[LatMin, LatMax]*, plages sur lesquelles nous pouvons travailler avec une tabulation plus fine que si nous avions gardé l'amplitude maximale de π.
Par défaut, nous nous en tiendrons à une tabulation dont le grain sera fixé à l'avance par un paramètre du programme.

Le programme côté serveur devra dont mettre en œuvre les deux optimisations décrites.

5.2.3. Visualisation interactive

Pour diffuser les cartes à un public large, il faut concevoir un client léger : facile à installer et utiliser. Nous avons choisi Java car il possède une richesse intéressante dans la bibliothèque graphique, et surtout il ne nécessite pas l'utilisation de *plugins* spécifiques comme SVG, ou bien Flash. Java nous permet aussi de déporter du côté du client une partie de l'intelligence de l'application puisqu'il nous permet de retravailler les résultats fournis par le serveur, De plus, nous exploitons ainsi l'expérience sur cette technologie du LIG.

La visualisation d'une carte lissée se fait usuellement en superposition avec un fond vectoriel présentant les limites administratives de l'espace lissé. Ces cartes présentent une gradation de couleurs, suivant l'intensité du phénomène, que nous souhaitons rendre paramétrable (choix de palette, type de progression). D'autre part, l'utilisateur doit pouvoir

jouer sur les paramètres suivants de l'analyse : les fonctions d'interaction, leur portée, la résolution, le cadrage (= le territoire ciblé) pour le lissage, et les indicateurs que l'on lisse. On veut en plus disposer de fonctions de zoom, déplacement, et de navigation dans l'atlas de cartes ainsi produit.
Nous décrivons dans le tableau 18 notre proposition concernant la collection cartes à produire :

Onglet	Contenu
1	Visualisation de l'aire d'étude et les contours d'unités administratives, plus une grille dont la taille des mailles correspondant à la résolution choisie pour le calcul.
2 3 4	image du potentiel calculé du numérateur (stock N) pour un voisinage v1 image du potentiel calculé du dénominateur (stock D) pour un voisinage v1 image du ratio des 2 précédents potentiels : Z = N/D sur le voisinage v1
5 6 7	image du potentiel calculé du numérateur (stock N) pour un voisinage v2 image du potentiel calculé du dénominateur (stock D) pour un voisinage v2 image du ratio des 2 précédents potentiels : Z = N/D sur le voisinage v2
8	Rapport ou différence entre Z(v1) et Z(v2)
9	Illustration d'un modèle de convergence économétrique indiquant combien il faut passer de N de v1 vers v2 pour que Z(v1) soit égal à Z(v2)

Tableau 18. Liste des cartes à produire.

Le format des données renvoyées par le serveur est contraint par les fonctionnalités attendues sur le client. Le client à tout avantage à récupérer la grille matricielle des valeurs calculées pour trois raisons.

Premièrement, le client doit pouvoir générer un rapport numérique (sous forme de fichier texte ou bien HTML) contenant les coordonnées géographiques de chaque point *M*, avec la valeur de son potentiel. Cette fonctionnalité qui existe dans le précédent module, *HyperAtlas*, est très utile car elle permet d'extraire des résultats de l'analyse pour par exemple les afficher dans d'autres logiciels. De ce fait, il était impossible que le serveur renvoie directement une image au format JPEG, PNG ou GIF, car on perd alors l'information numérique.

Deuxièmement, l'interactivité sur le choix des palettes, du nombre de classes et du type de progression dans la distribution des couleurs incite à garder les valeurs de potentiel calculées côté client : en cas de sélection de palette graphique ou de distribution différente, le client n'a pas à réémettre une requête en direction du serveur. Il peut ainsi recalculer une image rapidement sans interroger le serveur en cas de changements de préférences graphiques. Dans l'autre cas, si le serveur cachait la précédente réponse, de manière à ne pas refaire les calculs (et les modalités de ce *caching* (gestion de cache) ne sont pas évidentes à mettre en place), on augmenterait alors le trafic réseau.

Troisièmement, le client pourra exploiter la réponse sauvegardée pour construire les cartes de ratio. Pour une résolution et un cadrage déterminé, si le serveur a renvoyé les cartes de numérateur et de dénominateur, la carte de ratio sur la même portée se déduit simplement par une division (onglet 4 et 7).

5.2.4. Proposition technique

Nous déportons la partie graphique avec la visualisation du côté d'un client Web Java, tandis que les calculs et le traitement lourd de données sont effectués sur un serveur accessible à distance, via le protocole SOAP avec sécurisation des échanges via SSL, (cf. figure 78). Le traitement des données côté serveur est optimisé de façon à produire des résultats intermédiaires dans des temps n'excédant pas quelques secondes.

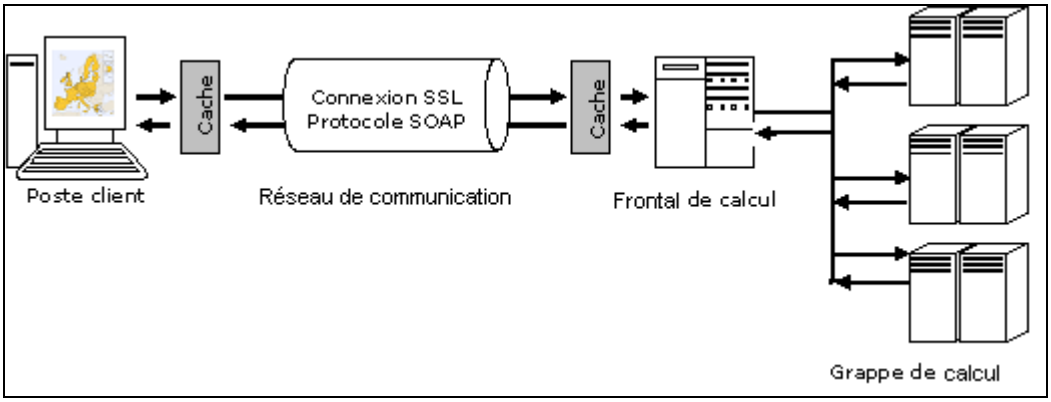


Figure 78. Vue générale de l'architecture.

5.3. Réalisation

5.3.1. Mise en oeuvre

Le serveur (écrit en C) et le client Java interagissent à travers une interface de communication, bien établie par le contrat WSDL. Ce contrat offre au client trois possibilités :

- Demander la liste des données disponibles sur le serveur, qu'il récupère sous forme d'un tableau de chaînes de caractères. On associe à l'ordre d'entrée du tableau un entier qui codifie le type de données sur lesquelles le client veut effectuer un lissage.
- Demander la liste des fonctions de lissage, qu'il récupère sous la même forme que les données, avec une codification par un entier des fonctions correspondant à l'ordre d'apparition de la fonction dans le tableau.
- Demander un lissage sur un jeu de données, avec un certain cadrage de la zone à lisser (les deux bornes sont exprimées en coordonnées géographique latitude/longitude), un type de fonction, sa portée et la résolution de l'image demandée (en nombre de points largeur par hauteur). Le serveur renvoie alors un tableau de flottants, non arrondis, correspondant à la grille calculée.

a) Structure de données du serveur

Nous détaillons ici comment les optimisations envisagées sur le serveur ont influencé sa structure de données.
Un *quadtree* est calculé par jeu de données, et à partir d'un découpage récursif de la zone d'information en quadrans, il conserve la somme des stocks des points inclus dans le quadran.

Les feuilles de l'arbre sont donc les coordonnées géographiques des mailles du jeu de données, valorisées par leur stock (cf. figure 79).

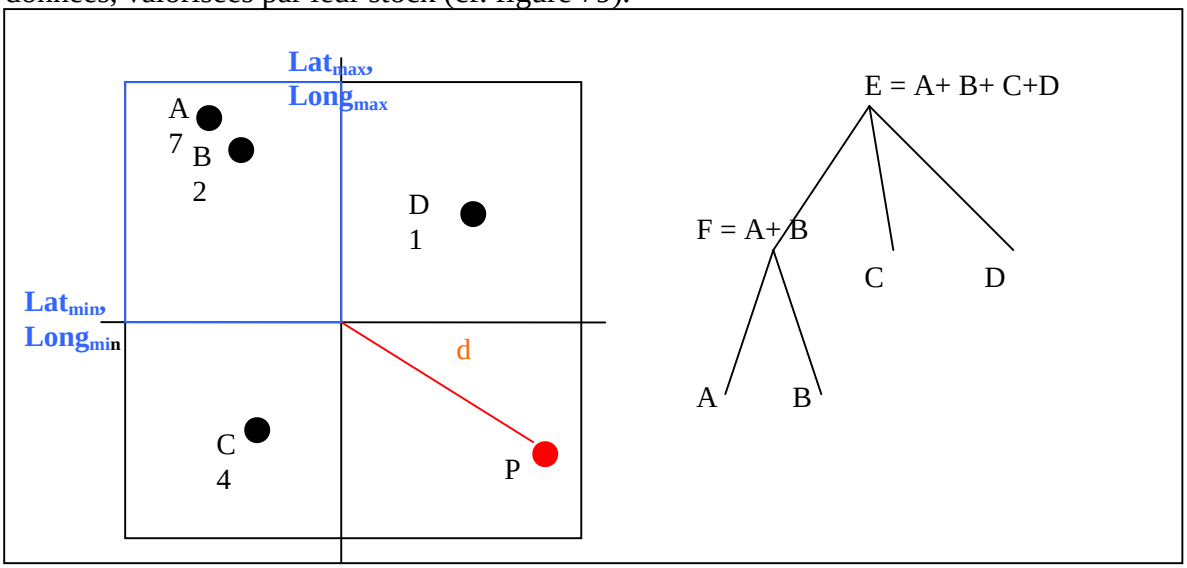


Figure 79. Découpage et utilisation du *quadtree*.

L'arbre conserve en mémoire la somme des stocks, mais il est impossible de pré-calculer les distances aux nœuds. En effet, cette distance varie en fonction du cadrage de la zone à traiter et de la résolution de la grille. A chaque calcul, on doit réévaluer la distance *d* entre chaque point P de la zone lissée et les nœuds de l'arbre. On teste alors si le produit de la distance *d* par la somme cumulée des stocks du sous-arbre est inférieur à une valeur seuil. En fonction du test, on abandonne la visite du sous arbre : un test positif signifie que la branche doit être ignorée pour le calcul.

Nous conservons aussi en mémoire un tableau de valeurs arccosinus calculées sur un intervalle fixe. Par contre, cosinus et sinus sont tabulées sur un intervalle qui dépend de l'amplitude en latitude ou longitude de la surface couverte par le jeu de données. En effet, la distance orthodromique entre 2 points A et B de coordonnées respectives A(latA, longA) et B(latB, longB) avec *r* rayon terrestre et les latitudes, longitudes en radians vaut :
d(A,B) = arccos(sin(latA) sin(latB) + cos(latA) cos(latB) cos(longB-longA)) * r.
Les valeurs de sinus à calculer vont varier sur l'intervalle des latitudes [latmin, latmax].
Les valeurs de cosinus varient elles à la fois sur l'intervalle des latitudes [latmin, latmax] et des longitudes [longmin, longmax]. Mais, sur ce second intervalle, elles ne sont pas pré-calculées pour l'instant : nous calculons cos(longB-longA) systématiquement.

A chaque nœud de l'arbre, nous associons donc la structure de données suivante :

- S : stock cumulé des fils
- jump : nombre des fils du sous arbre +1, (=1 si feuille de l'arbre) ou -1 si fin
- cos(latmin), sin (latmax), cos(latmax), sin (latmin), longmin, longmax (si feuille, alors longmin = longmax) : valeurs permettant d'évaluer la distance d.

Cette structure est ensuite conservée dans un tableau de structures (non pas de pointeurs) pour conserver la localité en mémoire, comme l'illustre le tableau 19.

Nœud	E	F	A	B	C	D
Jump	4	3	1	1	1	-1
Stock S	14	9	7	2	4	1

Tableau 19. Structuration des données sous forme de tableau.

La valeur du *jump* (saut) permet de parcourir le tableau efficacement. Par exemple, on parcourt le tableau avec un indice *i* et si le produit *d*S* est inférieur à la valeur du seuil d'élagage, alors *i* progresse de la valeur du *jump*. La valeur du seuil est paramétrable via un fichier de configuration. Elle vaut par défaut un millième de la masse totale des stocks.

b) Interface du client

Pour l'aspect de l'interface graphique, nous nous sommes inspirés de l'aspect de la partie graphique existant déjà sous *HyperAtlas*. Nous avons conservé un système d'affichage des cartes produites sous forme d'onglets, chaque onglet correspondant à une analyse spécifique.

Chaque onglet est séparé en deux parties. A droite, l'espace graphique occupe les trois quarts de la largeur et contient l'image lissée (cf. figure 80), avec son titre, une échelle, une barre pour zoomer. L'image matricielle que nous composons à partir de la grille de potentiels calculés est plaquée sur un fond vectoriel représentant le maillage administratif et délimitant les bordures côtières et de l'aire d'étude. En passant la souris sur la carte, l'utilisateur peut se déplacer dans la carte, et réduire ou agrandir la zone visualisée : ce qui a pour effet de relancer le calcul d'une nouvelle grille. En effet, la zone de visualisation est corrélée avec la zone à traiter.

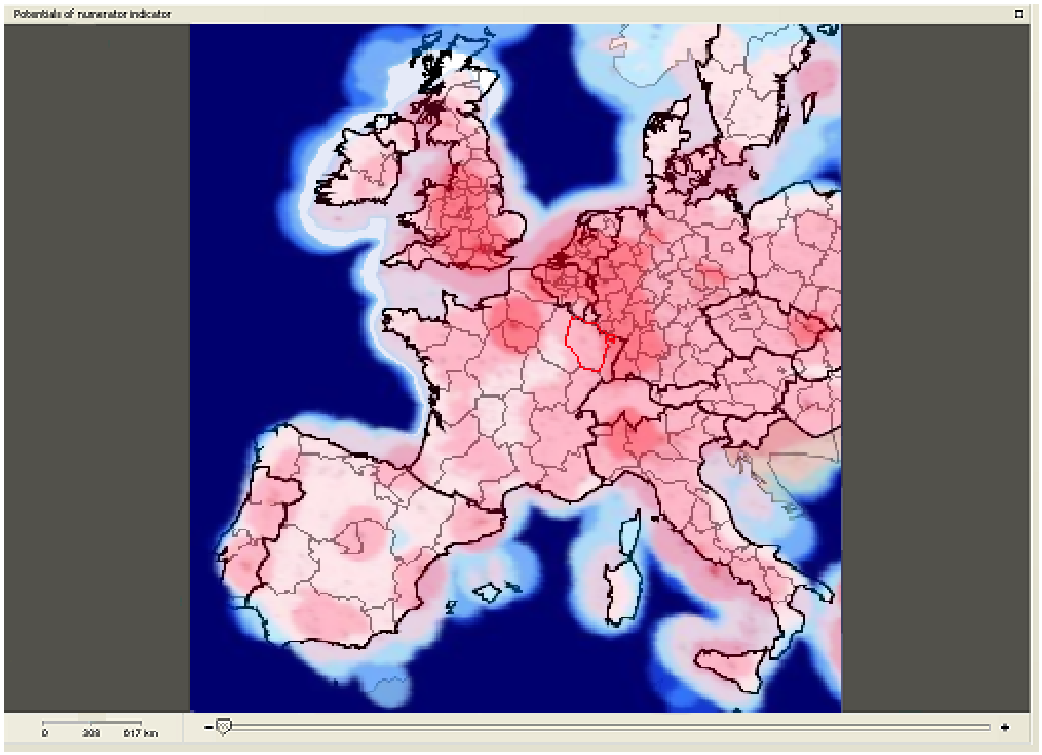


Figure 80. Contenu d'un onglet : carte de population lissée par une fonction gaussienne sur un rayon de 50km, sur l'Europe des 27, en résolution 300 par 400.

A gauche, un ensemble de 3 onglets dédiés à chaque carte : ils affichent la légende de la carte (cf. figure 81), des options pour le choix de la palette de couleurs et le mode de distribution des couleurs - nombre de classes de couleurs et progression dans chaque classe – (cf. figure 82), et enfin une explication de la sémantique de l'analyse présentée dans l'onglet sélectionné.

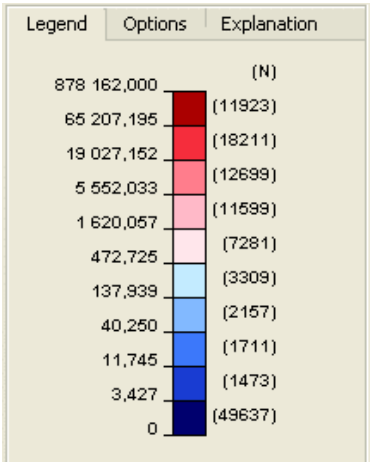


Figure 81. Légende de la carte

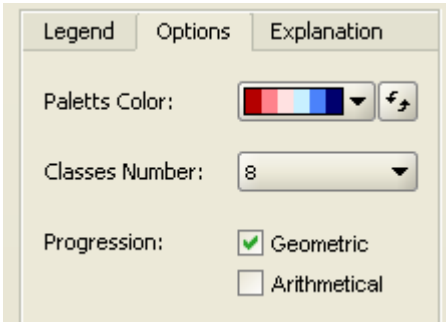


Figure 82. Options pour les couleurs et le mode de distribution de la carte

L'utilisateur peut agir sur les paramètres suivants de l'analyse :

- la fonction d'interaction : une liste déroulante présente la liste des fonctions implantées sur le serveur, au format textuel que ce dernier retourne au client.
- la portée moyenne en kilomètres de l'analyse, sélectionnée via une barre de glissement, ou par saisie numérique. Ce choix n'est pas borné.
- la résolution définie en nombre de points en largeur et hauteur désirés pour la grille, et qui se rapporte au pas de discrétisation de l'espace étudié.
- le cadrage (= le territoire ciblé) pour le lissage, correspondant actuellement à l'espace visualisé. Un zoom ou un déplacement modifie donc le cadrage.
- les stocks proposés dans deux listes déroulantes séparées, une pour le numérateur, l'autre pour le dénominateur, afin de pouvoir ensuite calculer un ratio de potentiel. Le client interroge le serveur lors de l'authentification de l'utilisateur pour connaître la liste des stocks disponibles (cf. figure 83).

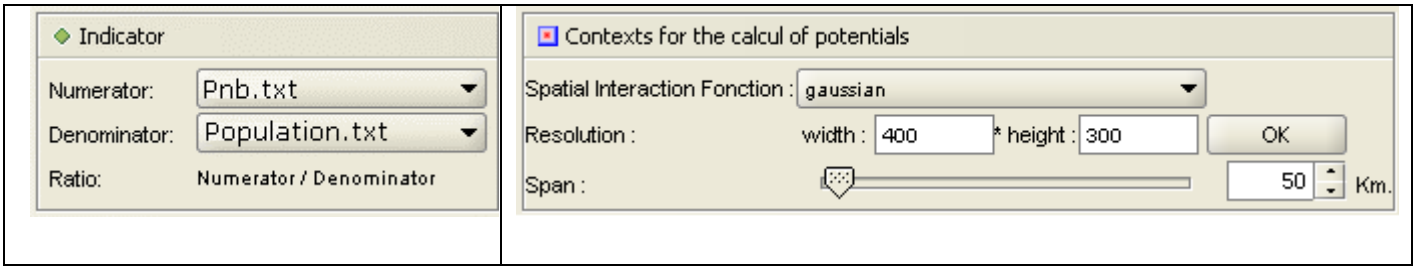


Figure 83. Liste des stocks et choix des fonctions, résolution et portée en km.

c) Architecture du client

La structure du client repose sur celle d'*HyperAtlas*, à la différence qu'il n'exploite pas la même couche logique. En effet, la logique (c'est-à-dire les calculs de ratio, de déviation globale, moyenne, et locale) n'est plus la même. Nous utilisons toujours les informations sur la géométrie et les maillages pour fabriquer un fond vectoriel, mais nous utilisons les calculs renvoyés par le serveur pour calculer une image matricielle. Ainsi la partie graphique qui peint une image dans un onglet est modifiée, ainsi que la partie accédant aux données représentées. Toutes les extensions de code sont regroupées dans un paquetage ***hypercarte.hypersmooth.***

Pour la partie graphique, nous avons l'idée d'exploiter la structure des onglets graphiques, et de construire chaque image via la superposition de couches graphiques. Pour implémenter cette superposition en couches, nous exploitons un objet déjà introduit par Christophe Chabert dans *HyperAtlas* : l'objet « *Layer* » [Chabert, 2007] qui joue le rôle d'un calque[36]. Par un mécanisme de tampon (*double buffering*), l'image est construite en mémoire par l'injection des différentes couches, avant d'être peinte dans l'objet graphique qui lui est dédié : la partie droite d'un onglet. Nous avons simplement enrichi l'objet *Layer* d'une méthode permettant d'ajuster son niveau de transparence. Le calque sert à composer une image par superposition et jeu sur les transparences (comme dans les logiciels de dessin et de retouche sur photo numérique).

Les couches graphiques se superposent dans l'ordre indiqué dans le tableau 20 et sont stockées dans une structure de hachage. La clé d'une couche est son nom, définit de manière statique, de manière à rendre le code et la manipulation de ces couches compréhensible.

Ordre de superposition	Description
5	Contour de l'unité sélectionnée
4	Contour des unités du maillage de plus haut niveau
3	Contour des unités du maillage sélectionné
2	Fond composé de mer et d'unités exclues de tout espace d'étude (Exemple : le Maghreb et la Russie sur le jeu de données Européen)
1	Image matricielle correspondant aux potentiels

Tableau 20. Superposition des couches (layers) pour construire une carte de potentiels

Comme nous n'utilisons plus la même logique pour l'affichage des cartes de potentiel, nous introduisons un nouvel objet dans le paquetage (*package*) dédié à l'affichage dans *HyperSmooth* : l'objet *SpatialAnalysisMap* dans le paquetage *hypercarte.hypersmooth.ui*. Cet objet va gérer :

- la superposition de couches pour construire une image
- la sensibilité aux évènements utilisateurs

Nous devons toujours traiter les évènements logiciels générés par l'interface graphique de façon différente par rapport à HyperAtlas : un zoom par exemple entraîne la demande

[36] Attention, le terme « calque » est employé habituellement en cartographie dans un sens différent : il représente alors une couche de données thématiques (réseau routier, fluvial, etc.) à superposer sur une carte.

d'une nouvelle grille de potentiels sur le serveur. Alors qu'avant on redessinait simplement l'image vectorielle sans recalculer aucune donnée.
Voici la liste exhaustive des évènements que SpatialAnalysisMap traite :

- zoom (agrandissement ou réduction),
- déplacement dans la carte,
- nouveau choix de donnée (stock),
- nouveau choix de fonction d'interaction,
- nouveau choix de résolution,
- nouveau choix de portée,
- changement d'options graphique : autre palette, autre mode de progression dans la discrétisation des couleurs, diminution ou augmentation du nombre de classes de discrétisation.

Excepté pour le dernier point de la liste précédente, tous ces évènements impliquent la demande d'une nouvelle grille de potentiels sur le serveur avec la méthode *launchGridComputing*(). Nous avons donc étendu la liste des évènements que traitait *HyperAtlas* en créant un paquetage dédié nommé *hypercarte.hypersmooth.event* contenant un objet *HSGlobalEvent* qui étend l'objet *GlobalEvent* existant auparavant dans *HyperAtlas*. Ainsi, nous profitons du code existant sans réécrire les mécanismes de gestion d'évènements.

Durant l'émission, l'attente et le traitement des requêtes au serveur, le client doit rester réactif, et proposer par exemple une fenêtre de dialogue indiquant que les traitements sont en cours : un sablier par exemple. Nous appelons ce composant *WaitDialog*.
En effet, le client s'adresse au serveur par des requêtes synchrones bloquantes. Elles permettent de récupérer la liste des fonctions d'interaction, et des données (stocks) disponibles sur le serveur, ou bien une grille de potentiels dépendant des paramètres qui sont fournis. Ces requêtes sont bloquantes, et on les exécute depuis le processus d'affichage de l'interface graphique (dont fait partie *SpatialAnalysisMap*), nous risquons de bloquer les interactions de l'utilisateur avec l'application. Le traitement des résultats renvoyés par le serveur sont donc délégués dans un objet dédié à la construction d'une image matricielle, *RasterImageManager,* qui utilise un **processus indépendant** pour lancer les requêtes sur le serveur (une instance de *SubmitGeoServerRequestTask*). Le diagramme de classe (cf. figure 84) fournit des explications plus détaillées concernant la composition de la classe *SpatialAnalysisMap* et ses interactions avec la classe *RasterImageManager*.

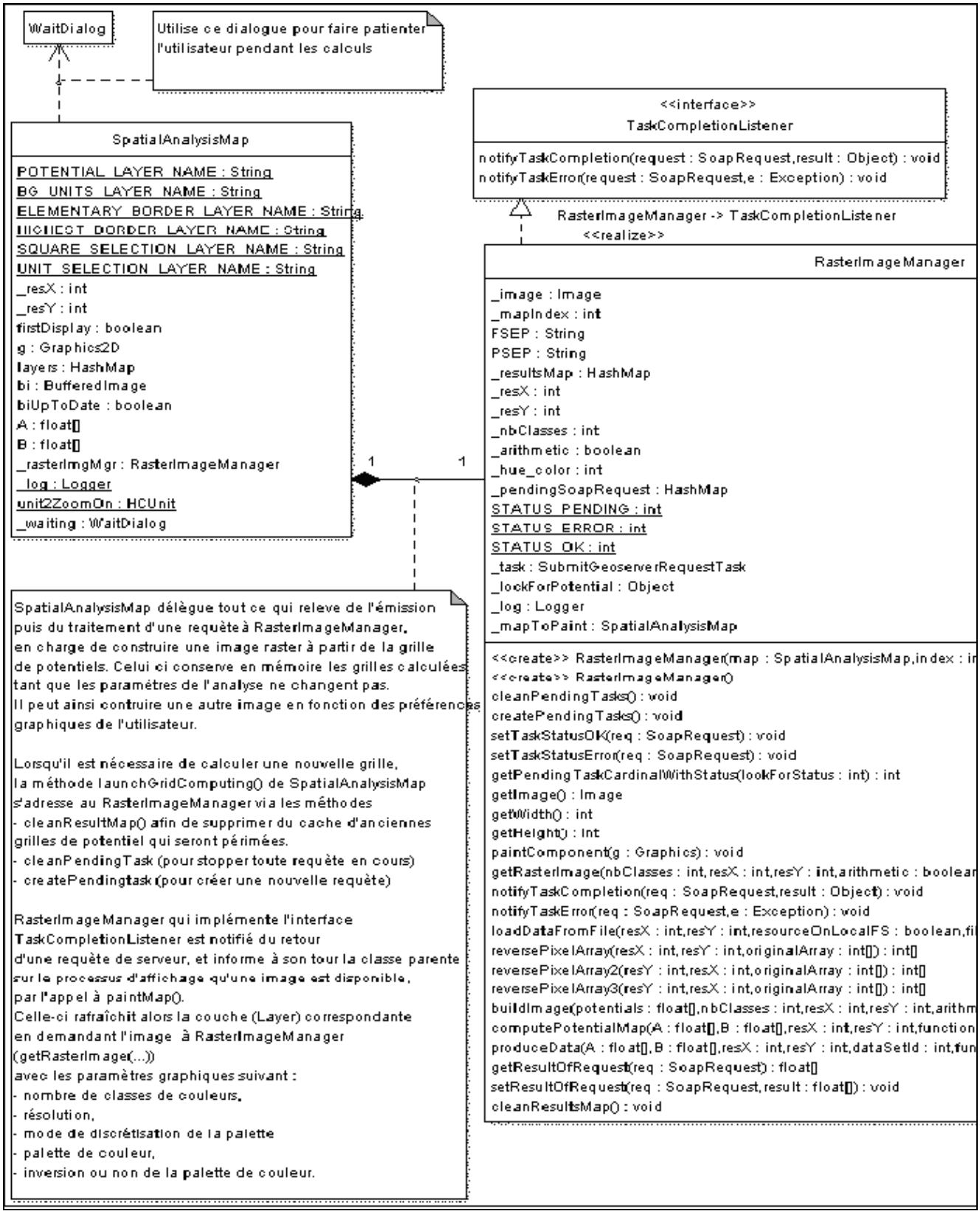


Figure 84. Diagramme de classe détaillant la composition de *SpatialAnalysisMap*.

La figure 85 montre le code de la méthode *launchGridComputing*() qui demande à l'objet *RasterImageManager* de créer une nouvelle grille de potentiel.

```
/**
 * Internal method used to prepare one/more request for computing a potentiel grid.
 * This must be called each time of the following parameters change :
 * - image bounds (after zooming or pan operation)
 * - spatial function id to apply
 * - data set id to read
 * - span (in km) of the potential function
 * - resolution in width (resX) or height (resY) requested for this image
 * Or when this index map is shown for the first time
 */
protected void launchGridComputing(){
    //update long/latitude of A and B in Settings
    this.updateCornerCoordinates();
    _waiting  = new WaitDialog("Loading image data, please wait a minute please !")
    _waiting.setLocation(this.getBounds().getLocation().x+this.getImageWidth()/2,
            this.getBounds().getLocation().y+this.getImageHeight()/2);
    _waiting.setVisible(true);

    _rasterImgMgr.cleanResultsMap();
    _rasterImgMgr.cleanPendingTasks();

    _rasterImgMgr.createPendingTasks();

}
```

Figure 85. Extrait du code de *SpatialAnalysisMap* : implémentation de *launchGridComputing*().

Le code que montre la figure 86 illustre la création d'une requête *SoapRequest* lorsque *SpatialAnalysisMap* sollicite *RasterImageManager* pour le calcul d'une nouvelle grille de calculs. On remarque que les paramètres d'analyse sont stockés et accessibles dans une classe *HSSettings*, qui complète la classe globale *Settings* de *HyperAtlas* avec l'ensemble des paramètres pour l'analyse spatiale de *HyperSmooth*. *HSSettings* observe le schéma de développement « singleton » [Gamma, 1999], (une instance statique unique) comme *Settings*.

```
/**
 * Ask for a new Image to be computed - Called by SpatialAnalysisMap
 *
 * It enqueues a list of soap requests to be completed by the same thread.
 * (this thread could be an instance of SubmitGeoserverRequestTask)
 */
public void createPendingTasks1() {

    List soapRequestList = new ArrayList();
    List myTaskListenerList = new ArrayList();

    ComputePotentialGrid req  = new ComputePotentialGrid();
    req.set_latitude_A(HSSettings.getInstance().getSpatialBounds_latitude_A());
    req.set_longitude_A(HSSettings.getInstance().getSpatialBounds_longitude_A());
    req.set_latitude_B(HSSettings.getInstance().getSpatialBounds_latitude_B());
    req.set_longitude_B(HSSettings.getInstance().getSpatialBounds_longitude_B());
    if (this._mapIndex == Settings.MAP_SPATIAL_ANALYSIS_NUMERATOR) {
        req.set_dataSetId(HSSettings.getInstance().getSpatialIndicatorNumeratorIndex());
        req.set_isNumerator(true);
    } else if (this._mapIndex == Settings.MAP_SPATIAL_ANALYSIS_DENOMINATOR) {
        req.set_dataSetId(HSSettings.getInstance().getSpatialIndicatorDenominatorIndex()
        req.set_isNumerator(false);
    }
    req.set_functionId(HSSettings.getInstance().getSpatialFunctionIndex());
    req.set_resolution_latitude(HSSettings.getInstance().getSpatialResolutionHeight());
    req.set_resolution_longitude(HSSettings.getInstance().getSpatialResolutionWidth());
    req.set_span(HSSettings.getInstance().getSpatialSpanValue());

    synchronized (_pendingSoapRequest) {
        _pendingSoapRequest.put(req, STATUS_PENDING);
    }
    // Add this request like a pending task -
    // Later, we could prepare 4 requests by example to compute an image gradually
    soapRequestList.add(req);

    //add myself like a listener for this task
    myTaskListenerList.add(this);

    //this is a non blocking call
    _task = new SubmitGeoserverRequestTask(soapRequestList, myTaskListenerList);
}
```

Figure 86. Code extrait de *RasterImageManager* : *createPendingTasks*()

Lors de la conception de ce client, nous avons essayé d'anticiper la mise en œuvre d'une gestion de flux de requêtes vers le serveur pour une construction et un affichage progressifs de notre image. Il suffira de modifier légèrement le code de *RasterImageManager* pour lui faire créer une liste de requêtes SOAP. Il est en effet conçu pour traiter une liste de requêtes, et un ensemble de grilles de potentiels reçus. Nous avons prévu à cet effet une table de hachage *_resultsMap* qui stocke et trie par une clé les grilles de potentiel reçues. Cette clé correspond exactement aux paramètres de la requête émise.

La requête est en effet modélisée par l'interface *SoapRequest* qui surcharge la méthode *equals* et *hashCode* afin de supporter le stockage en tant que clé dans une table de

hachage, suivant la recommandation 8 de Bloch [Bloch, 2002]. Cette interface est implémentée par 3 objets : *GetFunctionList* liste des fonctions d'interaction disponibles sur le serveur, *GetDataList* énumère les indicateurs que l'on peut traiter, et enfin *ComputePotentialGrid* demande une grille de potentiel au serveur, dépendant des paramètres de résolution, portée, fonction, etc., qui font partie des champs de l'objet.

Ces requêtes sont traitées par un processus indépendant *SubmitGeoserverRequestTask*, instancié par *RasterImageManager*. Ce processus s'adresse au serveur (*HypersmoothServiceImpl*) pour exécuter les requêtes bloquantes, et lorsqu'elles retournent un résultat, il notifie ses abonnés (ceux qui implémentent *TaskCompletionListener*) du résultat. Le diagramme de séquence de la figure 87 illustre cet enchaînement d'actions lorsqu'un utilisateur a sélectionné une nouvelle fonction d'interaction par exemple.

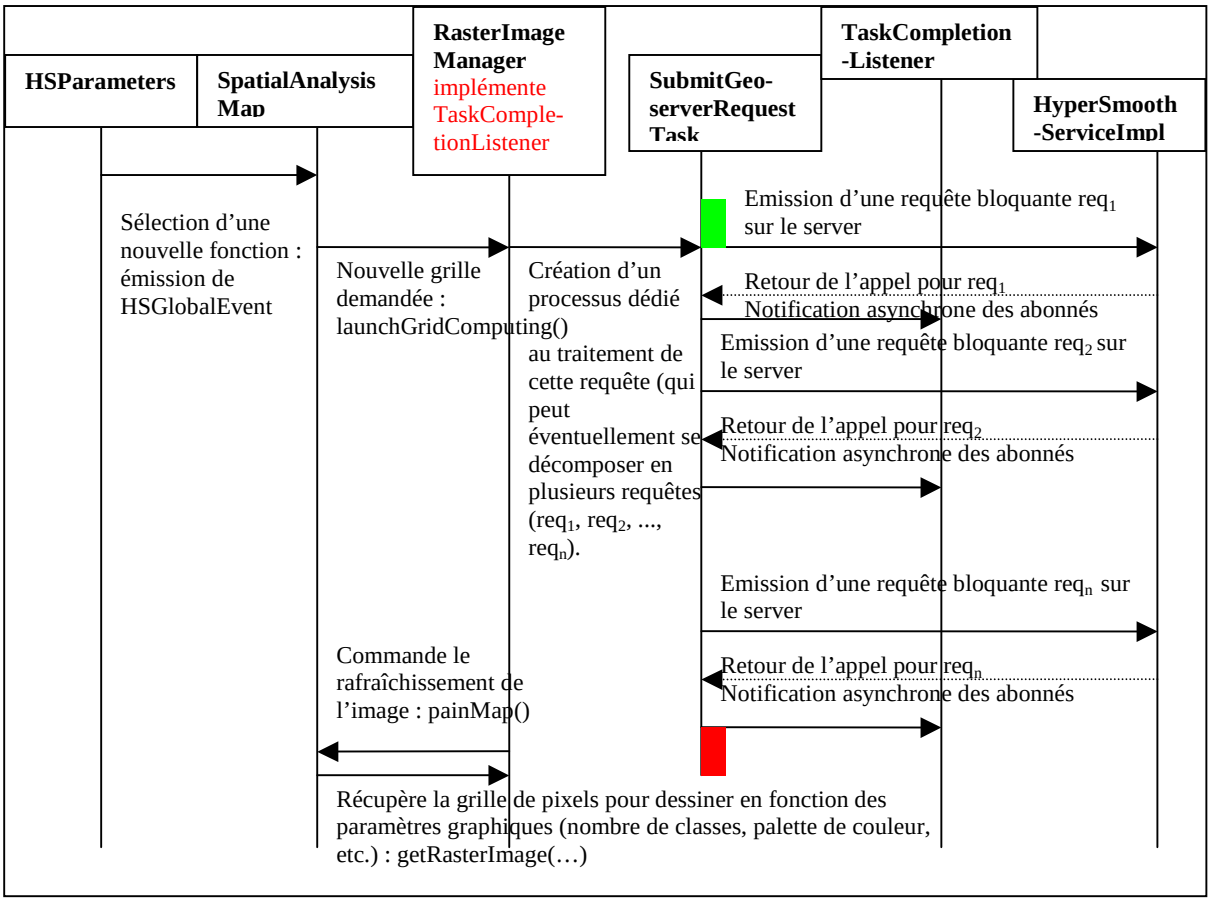


Figure 87. Diagramme de séquence d'actions entre composants pour dessiner une carte de potentiel.

5.3.2. Tests et performances

Le serveur peut s'utiliser de façon autonome par rapport au client, et nous mesurons alors les temps de calcul de grilles indépendamment de leur transmission sur le réseau, et leur représentation. Les performances présentées dans le tableau 21, qui sont plutôt bonnes, sont enregistrées sur une machine mise à notre disposition par la MSH[37], un octo-processeurs de

[37] http://www.msh-alpes.prd.fr/Vie/Axes2/Meth-Calcul.htm

type Altix 350 de marque SGI à mémoire partagée. Elle comporte 8 processeurs Itanium-2 (64 bits) bénéficiant d'une mémoire vive (8 Go) globalement partagée, et fonctionne sous Linux/SUSE Enterprise Server 9 (ia64) Version 9.

Nombre de valeurs de stock	Résolution	Portée (km)	Temps de calcul (s)
116203	400 x 300	100	35
288	400 x 300	100	9
116203	800 x 600	100	165
116203	400 x 300	50	15

Tableau 21. Temps de calcul des matrices sur le serveur, avec une fonction gaussienne.

Il faut retenir qu'à l'avenir, ce programme pourrait être exécuté par une machine multi-processeurs de type SMP, avec mémoire partagée, et dès lors, les temps de calculs cités seraient divisés par 4, 8, ou plus, suivant le nombre de processeurs mobilisés.

Les temps de latence réseau sont bons, malgré le fait que nous n'avons pas encore mis en œuvre la compression des données échangées via gzip. En fait, on mesure un temps correspondant à l'emballage de la réponse, l'encryptage et le déballage de 4 s localement avec une résolution de 300 * 400.
Côté client, la reconstruction de l'image à partir d'un tableau de flottants prend très peu de temps (128 ms). Cette matrice est demandée dès lors que l'utilisateur change un des paramètres du lissage, ou bien agrandit, réduit ou se déplace dans la zone de visualisation. La matrice reçue est mise en cache, avec les paramètres de la requête correspondante afin de pouvoir retravailler dessus ultérieurement. La résolution demandée étant le plus souvent inférieure à celle de l'image vectorielle (1027 * 688), nous appliquons une interpolation bi-quadratique sur l'image produite afin de la caler sur le fond vectoriel.

En revanche, du côté de la visualisation du client, nous n'obtenons pas une image véritablement satisfaisante : d'une part parce que la superposition d'une image matricielle sur un fond vectoriel déjà coloré ne donne pas un bon rendu. D'autre part, le cadrage de l'image vectorielle sur l'image matriciel n'est pas excellent. Il s'améliore en zoomant sur des pays, et ce problème vient du calcul des bornes du cadre.

5.4. Bilan

L'architecture dans son ensemble fonctionne, et nous répondons aux critères à la fois d'interactivité et de confidentialité que nous avions fixés. Côté serveur, nous obtenons une grille de potentiels exploitable dans des temps satisfaisants. Bien que la qualité de la visualisation côté client puisse être améliorée, le niveau d'interactivité du client est en adéquation avec nos besoins.

5.4.1. Perspectives d'amélioration

Nous donnons ici une liste assez conséquente d'améliorations et d'optimisations auxquelles nous avons réfléchi et qui sont aussi le fruit de nos réunions d'équipe. Il serait dommage de perdre ce travail, c'est pourquoi nous retranscrivons toutes les idées.

a) Possibles optimisations pour le serveur

Plusieurs pistes s'offrent pour améliorer les performances du serveur de calcul. Par exemple, le calcul de la distance orthodromique peut bénéficier d'un pré-calcul supplémentaire. De même, un gain de bande passante serait obtenu par la réduction de la précision des valeurs transmises par le serveur au client.

- Tabuler plus encore

Nous tabulons déjà les formules de cosinus et sinus pour le calcul des distances orthodromiques, mais il reste encore possible d'optimiser le calcul de *cos(lonB-lonA))* en utilisant la formule suivante et en tabulant les valeurs de cosinus et sinus en longitude :
Cos (longB – longA) = cos (longB) cos (longA) + sin (longB) sin (longA)
Il s'agit aussi de trouver un bon équilibre entre l'espace mémoire ainsi consommé et le gain de temps obtenu. Si la tabulation est trop grossière, la précision sur les résultats du calcul s'en fera ressentir.
Les paramètres influençant le niveau de finesse de la tabulation sont les suivants :
- aire de la surface totale lissée : plus la surface est grande, plus on peut se permettre une approximation grossière.
- portée du lissage : pour une petite portée, il vaut mieux tabuler finement
L'idéal serait de déterminer quel lien théorique lie les deux paramètres. Par ailleurs, la recherche du bon compromis espace mémoire/gain de temps est une préoccupation des programmeurs de systèmes embarqués ou de processeurs numériques de signal (usage intensif des transformées de Fourier sur fonction trigonométrique). De nombreuses études existent sur ce sujet, dont nous pourrions nous inspirer.

- Tronquer les valeurs de potentiel

La troncature (c'est-à-dire la réduction de la précision) des flottants transmis dans la grille matricielle réduirait le volume des données échangées, et nous accorderait donc un gain de temps sur la transmission des données. Mais il faut comme pré-requis étudier les plages de variation des valeurs, pour déterminer quelle précision est suffisamment discriminatoire. Spontanément, nous soupçonnons que cette plage de variation est dépendante à la fois du jeu de données, de la fonction utilisée, et du niveau de résolution, et varie donc à chaque requête. Un algorithme qui automatise la détermination du niveau de troncature est en cours d'élaboration.

D'autres pistes, moins classiques, sont liées à l'usage de notre algorithme de cut-off. Il s'agit de préparer une stratégie de sous-échantillonage des données.

- Adapter la valeur du seuil d'élagage dynamiquement

La valeur du seuil à partir duquel nous élaguons les données dans le *quadtree* influence la durée du calcul et l'exactitude des résultats : un seuil bas rallonge le temps de calcul. On peut fixer arbitrairement sa valeur à 1/1000ème de la masse totale des stocks, ou bien adapter le seuil pour chaque point de la grille considéré, et moyenner le stock sur simplement un certain voisinage du point. Mais ce voisinage peut dépendre de la fonction d'interaction ou de la portée utilisée. Ce seuil peut aussi être défini en sommant les stocks sur un périmètre défini par la zone d'analyse, plus une bordure correspondant peut-être à la portée d'analyse.

- Sous échantillonner les données du serveur

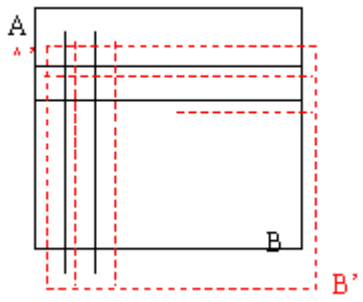
On peut se contenter d'un échantillon de données plus généraliste en limitant la visite de l'arbre à un certain niveau, sans aller jusqu'aux feuilles, lorsque que le niveau d'information requis est très grossier. Cette simplification permet de donner une vue générale au client en fonction d'un certain rayon de lissage et de la fonction choisie. Ce filtrage avant les feuilles limite le nombre de calculs de distance. De plus, en cas de zoom plus fort, l'arbre reste calculé et utilisable. Mais il faut déterminer à partir de quelle surface minimum, on peut élaguer le dernier niveau de l'arbre pour faire le calcul de potentiel. Est-ce que cette surface dépend du jeu de données, de la portée, de la fonction d'interpolation ? Ceci est à vérifier et nous devons établir une stratégie du côté du serveur pour établir cette surface limite.

- Sélectionner le type de distance dynamiquement

Enfin, la sélection de la distance pourrait se faire de manière adaptative, selon l'échelle d'étude retenue : si la zone d'analyse présente une amplitude en latitude et longitude suffisamment petite (par exemple, la région Rhône-Alpes) on utiliserait la distance euclidienne sans introduire de déformations notables. Ceci optimiserait la vitesse de calcul du serveur, et donc permettrait de satisfaire des demandes de résolutions plus élevées. Il incomberait au serveur de décider si la zone de lissage demandée peut se contenter d'une distance euclidienne (avec un test de surface couverte).

b) Possibles optimisations pour le client

- Superposer des grilles calculées :



On peut, à partir d'une image calculée sur une zone (A,B), avec une certaine définition, effectuer un décalage sur les pixels en calculant une seconde image ayant la même définition mais sur une zone (A', B'), à condition de décaler légèrement les bornes du calcul A' et B'(cf. figure 88). Le résultat est une image de résolution doublée, qui anticipe zoom et déplacements à l'intérieur de cette zone. Cette tâche revient au client car elle nécessite la mémoire de l'image précédemment calculée, et la connaissance du niveau de zoom.

Figure 88. Superposition de 2 grilles de pixels pour la constitution d'une image.

- Combiner la superposition de grilles et l'interpolation de pixels

Le client a aussi la possibilité sur une zone donnée de réitérer 4 fois sa requête tant que l'utilisateur n'envoie pas de nouvelle requête. Il effectue une interpolation sur la première image de résolution faible. Puis complète en demandant 3 fois la même image à résolution identique, mais en décalant un peu les bornes à chaque fois. Cette technique à l'avantage de raffiner l'image au fur et à mesure, tout en se préparant à un zoom de l'utilisateur. Dès que l'utilisateur demande le lissage sur une zone différente de la carte, le client adresse des nouvelles requêtes prioritaires sur le serveur, et abandonne le traitement de sa liste de requêtes en cours.

- Calculer une image vectorielle

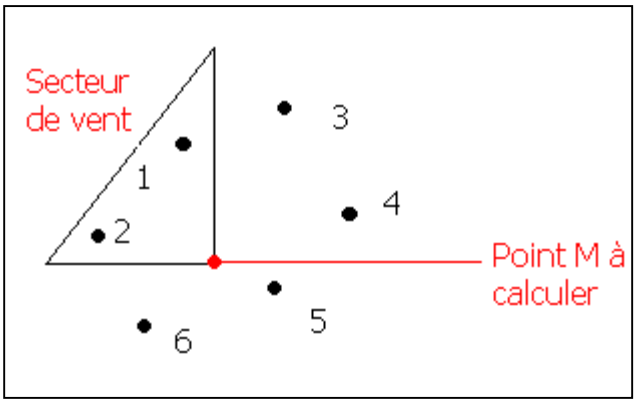
Etant donné le mauvais rendu d'un placage d'image matricielle sur fond vectoriel, nous aurions intérêt à tenter de calculer des courbes de niveau sur la matrice (en fonction du nombre de classes de couleurs exigé par l'utilisateur). Les images de potentiel présentant souvent des gradations de couleurs et des zones de discontinuités non aléatoires, cette technique de vectorisation de l'image peut s'appliquer facilement. Nous obtiendrions ainsi un meilleur rendu visuel.

- Permettre une sélection libre de la zone à lisser et contrôler l'émission de requêtes

La zone de traitement est celle visualisée par l'utilisateur dans un onglet : par conséquent, tout zoom ou déplacement implique une nouvelle requête émise sur le serveur. Or nous désirons décorréler ces actions, et il serait judicieux d'une part que l'utilisateur puisse sélectionner une zone particulière à traiter par un cadre de sélection, et d'autre part, que le calcul soit demandé par une action explicite de l'utilisateur sur un bouton de soumission, lorsqu'il est certain des paramètres choisis pour son analyse.

5.4.2. Evolution de la méthode

Nous proposons ici la possibilité d'utiliser des voisinages anisotropiques, fondés sur les secteurs angulaires. Joël Boulier est à l'origine de cette adaptation de l'approche du potentiel [Boulier, 2003]. En travaillant sur des phénomènes de tempêtes, aux flux très orientés, il a développé autour de la méthode générale des potentiels une vision anisotropique de l'espace. Ici, on ne considère plus le phénomène observé comme une mesure dans un certain voisinage isotropique mais comme une énergie potentielle vers ce lieu, construite par la mise en relation au lieu de phénomènes proches sous condition d'orientation origine-destination.
Ce développement met en avant, par exemple, la relation amont-aval, si importante quand on étudie les impacts de flux (tempêtes, hydrologie, etc.). L'idée forte est de promouvoir les facteurs perturbants qu'un flux pourrait rencontrer avant d'atteindre un lieu. Pour quantifier le potentiel en un lieu, nous tenons compte à la fois de la distance de la source d'information au point à estimer, mais aussi de son orientation dans l'espace.
A titre d'exemple, seuls les points connus 1 et 2, dans la figure ci-dessous, seront conservés dans le calcul du potentiel.



Cette modification de la méthode générale repose sur une hypothèse forte : l'environnement spatial d'un point n'est pertinent que selon un secteur déterminé par la thématique (vents dominants, courant marin,…). Les corrélations entre les potentiels ainsi déterminés et les dégâts constatés sont très nettes.

Figure 89. Prise en compte des secteurs de vent.

5.4.3. Calendrier effectif des réalisations

Nous présentons ici un historique des réalisations afin d'illustrer la répartition et le déroulement des tâches que nous avions définies entre l'équipe MESCAL, dont Serge Guelton fait partie, et l'équipe STEAMER.

Benoît Vaultier est à l'origine du premier code C du serveur conçu pour fonctionner de façon autonome, et mettant en œuvre le *cutoff* algébrique. Serge Guelton, embauché en mars 2006 a conclu un contrat de travail dans l'équipe MESCAL s'étalant du 15 mai au 15 septembre. Durant ces cinq mois, sa mission portait sur la mise en place d'une une plate-forme de calcul distribuée utilisant le code C existant. Nous devions nous synchroniser pour concevoir une architecture intégrant l'utilisation d'un client graphique sous la responsabilité de l'équipe STEAMER.

Nous exposons la liste des tâches définies dans le projet *HyperSmooth* (cf. figure 90), ainsi qu'un calendrier des travaux réalisés durant ces cinq mois, sous la forme d'un diagramme de Gantt (cf. figure 91).

A l'issue de cette période, du 15 septembre au 15 octobre, nous avons continué le développement du client. Ces développements seront repris le 15 octobre par Raphaël Thomas, qui débute un mémoire CNAM au sein du projet HyperCarte sous la direction de Jérôme Gensel. Nous avons consacré deux semaines environ à lui transférer nos connaissances.

ID	Nom	Début	Fin
1	Etat de l'art - méthode du potentiel	15/03/2006	30/03/2006
2	Couplage avec le serveur par un protocole	15/05/2006	15/09/2006
2-1	Recherche et test de protocole reseau adapté	15/05/2006	15/06/2006
2-2	Implémentation d'une interface de communication coté client	15/06/2006	30/06/2006
2-3	Tests de validation du couplage	28/08/2006	15/09/2006
3	Conception et réalisation d'un client graphique pour HyperSmooth	15/05/2006	11/10/2006
3-1	Visualisation d'une carte de potentiel dans une fenêtre simple, à partir d'un fichier de données	15/05/2006	30/05/2006
3-2	Visualisation d'une carte de potentiel dans un onglet, exploitant l'interface graphique de HyperAtlas	23/08/2006	25/08/2006
3-3	Gestion d'un paramétrage dynamique d'une carte	28/08/2006	15/09/2006
3-4	Gestion de plusieurs cartes, et d'une liste de requètes bloquantes, d'un cache en mémoire	15/09/2006	11/10/2006
4	Encapsulation du serveur C, et paramétrage	15/05/2006	23/08/2006
4-1	Analyse du code	15/05/2006	31/05/2006
4-2	Amélioration et débuggage du code : tabulation, gestion de cache, configuration par paramètres du serveur	01/06/2006	30/07/2006
4-3	Préparation des distributions du serveur pour un OS Linux	31/07/2006	23/08/2006
4-4	Implémentation d'une interface de communication coté serveur	15/06/2006	30/06/2006
CP	Plumejeaud	15/03/2006	31/01/2007
SG	Guelton	15/05/2006	15/09/2006

Figure 90. Liste des tâches définies dans le projet *HyperSmooth*.

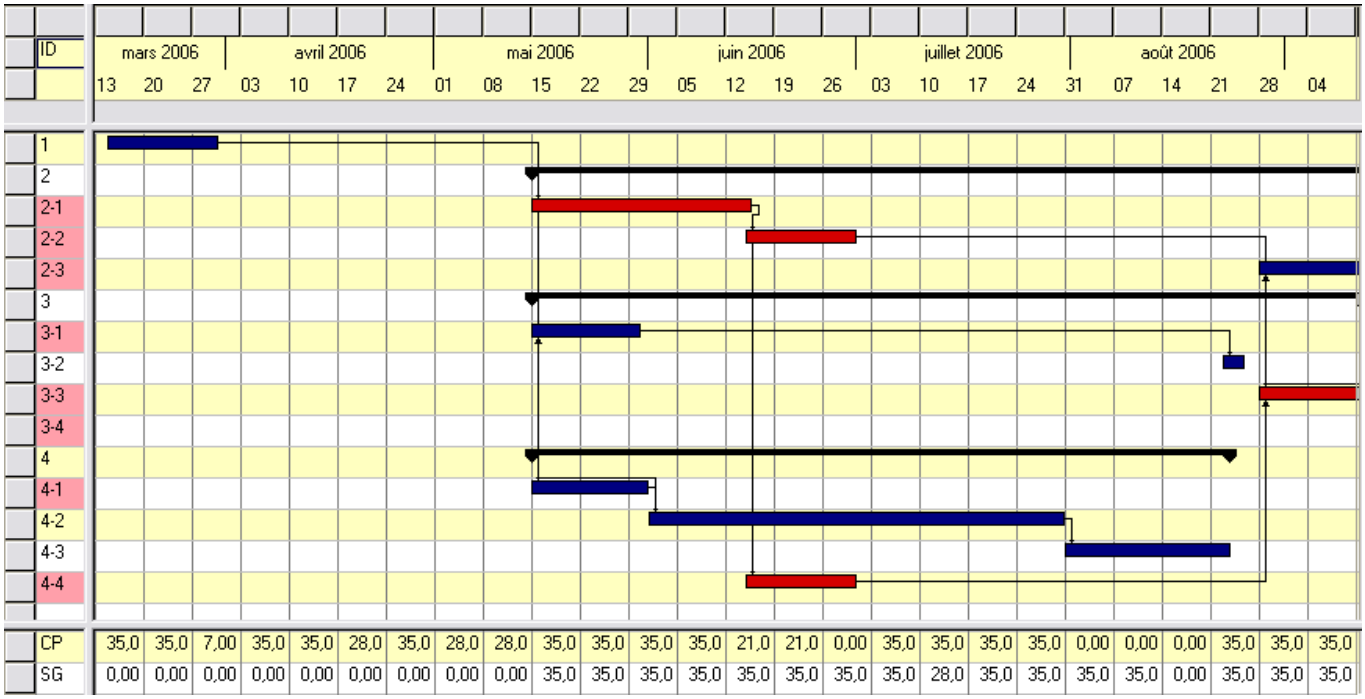


Figure 91. Calendrier de réalisation des travaux sur *HyperSmooth*.

6. Synthèse et perspectives

Ce chapitre conclut le mémoire par un rappel des objectifs, et le bilan établi en fonction de ceux-ci. Une ouverture vers les perspectives qu'offre ce travail est proposée, ainsi qu'un bilan personnel sur l'expérience vécue durant ces douze mois de stage.

6.1. Rappel des objectifs

Dans le cadre d'un groupe de recherche pluridisciplinaire nommé *Hypercarte*, entre informatique et géographie, une application cartographique pour l'analyse territoriale multiscalaire avait été développée. L'application *HyperAtlas* est reconnue par une large communauté d'utilisateurs et un organisme européen, ESPON, pour l'observation et l'aménagement du territoire européen.

L'application, suffisamment mature pour être déposée à l'APP, ne fait l'objet que de quelques demandes de maintenance évolutive de la part de ESPON, dont nous sommes en charge. Cependant, elle présente aussi deux lacunes assez évidentes : elle n'offre aucun moyen pratique et convivial pour une acquisition des données autonome, et elle ne propose aucune forme de représentation continue à l'utilisateur, qui reste prisonnier des maillages territoriaux dans son analyse.

Nos objectifs visaient à combler ces deux lacunes, et notre proposition consistait à conduire le développement de deux modules applicatifs, indépendants de *HyperAtlas*, mais dont une partie du code et de la conception graphique dérive du premier logiciel. De plus, afin de rationaliser les efforts fournis, nous nous sommes focalisés sur la réorganisation du projet, pour rendre le code plus facilement maintenable, et le suivi des différentes versions plus aisé.

Le module *HyperAdmin* doit permettre aux utilisateurs de créer leurs propres jeux de données pour *HyperAtlas*, dans son format spécifique. Cette intégration autonome de données fournies sous forme de fichiers ASCII, et qui était jusqu'alors dévolue aux spécialistes et à l'ingénieur du projet, élargira le public *d'HyperAtlas* et son intérêt en multipliant les jeux de données. Le cahier des charges spécifie aussi que l'on puisse pré-visualiser ses données avant de générer les fichiers sérialisés pour *HyperAtlas*. On doit aussi pouvoir modifier les stocks, voire même les maillages, les hiérarchies, etc., depuis l'interface graphique, et aussi attribuer des champs descriptifs aux statistiques étudiées. D'autre part, une demande très importante concerne le support de matrice de distances temps voiture et camion dans *HyperAtlas*, et elle impacte profondément le modèle de données, qui est généré par *HyperAdmin*.

Quant au module *HyperSmooth*, il doit mettre en œuvre une méthode originale pour la représentation continue de phénomènes sociaux-économiques : la méthode du potentiel. Elaborée depuis plusieurs années au sein du projet, elle repose sur un modèle mathématique adapté pour des données aux propriétés additives, et récoltées sur des maillages spatiaux, telles que celles employées par *HyperAtlas*. Cependant, le calcul des potentiels d'une carte exige de puissantes ressources de calcul, car celui-ci est coûteux. Une architecture distribuée doit être conçue, avec d'une part un client léger présentant une interface graphique très interactive pour le paramétrage de l'analyse, et d'autre part un serveur de calcul connecté sur une grappe de PC, offrant une liaison sécurisée avec le client, par authentification.

6.2. Synthèse

Nous avons jeté les bases d'un développement durable pour toute la plateforme logicielle du groupe de recherche Hypercarte. En effet, nous avons mis en place des outils de gestion de configuration et de suivi de projet, avec l'InriaGforge. Nous avons essayé d'augmenter la maintenabilité du code des trois modules en harmonisant les méthodes de développement : création d'un système de journalisation des applications efficace et configurable, centralisation des traductions des messages applicatifs dans des fichiers de propriété, par exemple.

En dehors de ce besoin général, nous avons aussi répondu aux besoins spécifiques exprimés par les utilisateurs avec le développement des modules *HyperAdmin* et *HyperSmooth.* En particulier, si toutes les fonctionnalités exprimées dans le cahier des charges ne sont pas toujours remplies, on peut estimer que leur architecture est aujourd'hui à même de servir de base pour les futurs développements, car ils sont conçus pour pouvoir évoluer.

HyperAdmin permet aujourd'hui aux utilisateurs d'acquérir leurs données de façon autonome. Il propose pour cela un menu Admin intégré dans l'interface graphique de *HyperAtlas*, sans duplication de code. L'utilisateur peut visualiser ses données dans *HyperAdmin*, de façon identique à *HyperAtlas*, avant de les exporter pour *HyperAtlas* dans des fichiers de données sérialisées. Il fabrique aussi un jeu de données qui intègre l'usage de matrices de distance entre unités, lues dans des fichiers, et introduit un concept nouveau pour *HyperAtlas*, celui de voisinage (autre que la simple contiguïté).

Le bilan de notre réalisation sur *HyperSmooth* s'avère positif quant au choix de l'architecture distribuée : répartition des calculs sur un serveur parallèle d'un côté, visualisation et paramétrage de l'analyse sur un client Java intelligent de l'autre, les deux parties étant connectées via un protocole de plus en plus répandu et présentant une accessibilité et sécurité maximale : SOAP, couplé avec un cryptage SSL. De plus les calculs sont accélérés grâce à l'utilisation d'une méthode de *cut-off* algébrique, et une tabulation des distances orthodromiques. Ce module fait par ailleurs l'objet d'une publication au colloque SAGEO 2007, [Plumejeaud et al., 2007].

6.3. Perspectives

La plate-forme logicielle dans son ensemble est un champ d'expérimentations pour les géographes et les informaticiens. Les deux nouveaux modules sont un terreau fertile pour les idées, qui ne manquent pas. HyperAtlas bénéficie aussi de cet enthousiasme renouvelé.

HyperAdmin qui doit être achevé pour offrir toutes les fonctionnalités proposées par le cahier des charges, sera décliné en différentes versions, destinée aux différents cercles d'utilisateurs de la plateforme. Les utilisateurs simples continueront d'utiliser HyperAtlas, tandis que les experts auront le droit d'utiliser une version d'HyperAdmin, où le menu Admin aura été amputé pour ne proposer que l'ajout et la modification de stocks dans les jeux de données, en travaillant directement sur le modèle de données en mémoire, sans connexion sur une base de données. La version « administrateur » ressemblera à celle d'aujourd'hui, et non

seulement tous les menus seront implémentés, mais aussi une sorte de « *wizard* » guidera les utilisateurs pour faciliter l'usage du logiciel, connectés sur une base de données.

D'autre part, quelques optimisations du modèle de données sont à prévoir pour permettre de manipuler une quantité importante d'unités territoriales. Le jeu de test validant cette optimisation serait la France des 36000 communes.

Un nouveau cas d'usage *d'HyperAdmin* pourrait concerner *HyperSmooth* : il pourrait générer des jeux de données pour *HyperSmooth*, à partir des fichiers ASCII actuellement fournis. En effet, *HyperSmooth* exploite des fichiers texte qui ont quatre entrées : un code d'unité territoriale, un couple (latitude, longitude) situant le centre de cette unité, et la valeur du stock pour cette unité, comme l'illustre le tableau 22.

Code unité	Latitude	Longitude	Stock
DE131	50	15	152000

Tableau 22. Format de données attendues pour Hypersmooth.

Concernant *HyperSmooth*, plusieurs pistes s'offrent pour améliorer les performances du serveur de calcul, que nous avons largement détaillées dans notre bilan sur ce module. Mais nous en reprenons ici les principaux axes : poursuivre sur la voie des précalculs de fonctions trigonométriques, mais surtout revoir la politique de réduction de somme et de calcul de distance pour mettre en place des stratégies adaptatives en fonction de la taille et de l'emplacement des zones couvertes par l'analyse.

Côté client, quelques améliorations pour le client sur le plan de l'ergonomie sont prévues. Comme, par exemple, décorréler la zone de visualisation et la zone de traitement, en utilisant un cadre de sélection de la surface à traiter. Nous souhaitons donner une interprétation plus directe de la résolution. Enfin, le calcul et la représentation de courbes de niveau à partir de l'image matricielle amélioreraient sensiblement le fondu avec le fond vectoriel représentant le maillage administratif. Le client offre aussi un champ de réflexions en ce qui concerne l'usage de caches autorisant un raffinement progressif des images, grâce à l'émission de requêtes successives.

Toujours à propos *d'HyperSmooth*, une question d'ordre théorique et algorithmique, porte sur l'introduction d'autres distances, et l'extension de la méthode de façon à illustrer les propriétés anisotropiques de l'espace géographique réel. Comme l'a montré une étude des phénomènes de tempêtes, aux flux très orientés [Boulier, 2003], il est intéressant de développer sur la base de la méthode générale une vision anisotropique de l'espace. Cette modification de la méthode générale permet d'intégrer de nouvelles formalisations des mouvements dans l'espace (vents dominants, courant marin, etc.) complétant celles qui sont associées à la fonction d'interaction spatiale.

La question du retour des usagers (*feedback*) et des effets induits sur la connaissance des utilisateurs se pose pour aborder le thème de la cartographie participative, mais n'est pas traitée. Il reste que nous gardons à l'esprit que la collecte des profils utilisateurs serait intéressante.

6.4. Bilan personnel

Mon bilan s'articule autour de deux perspectives : professionnelle et personnelle. Quels profits puis-je tirer d'une telle expérience, mais aussi comment est-ce que je l'ai ressentie et vécue ?

6.4.1. Sur le plan thématique et technique

Le sujet de mon mémoire concorde avec le thème que j'avais choisi d'aborder pendant mon probatoire : la cartographie interactive. Il ajoute de plus un aspect analyse spatiale sur des thèmes sociaux ou économiques que j'apprécie beaucoup. En effet, j'aimerais à l'avenir travailler dans le domaine de la géomatique, pour contribuer à la diffusion de la connaissance de notre géographie et notre environnement. Il existe de nombreux domaines d'application de mes connaissances acquises qui deviennent des enjeux sociaux et font même partie des thèmes de la campagne présidentielle française actuelle, comme par exemple, la gestion rationnelle de l'environnement et des ressources naturelles pour un développement durable. Ou bien encore la problématique des flux migratoires et les questions de ré-équilibrage des richesses à toutes les échelles (continents, pays, régions) qu'elle soulève.

Techniquement, j'ai découvert le domaine des services Web avec l'utilisation du protocole SOAP, et j'ai approfondi mes connaissances sur les réseaux, et la sécurisation des échanges sur un réseau. J'avais déjà en effet un bagage de connaissances techniques en matière de développement plutôt important. Je pratique Java depuis 6 ans, ainsi que le MVC via Struts, et j'ai de l'expérience en matière de développement Web (servlets, JSP, HTML). Ces connaissances, ainsi que les méthodes d'industrialisation comme la gestion de configuration et l'utilisation de Ant, je les ai mises au service du projet pour le faire progresser sur le plan de l'organisation. La difficulté technique résidait principalement dans la définition du cahier des charges des deux modules, et la conception d'une architecture adaptée. D'ailleurs, ce sont de ces choix dont j'ai le plus douté, jusqu'à enfin leur concrétisation dans chacune des réalisations. Finalement, au vu du bilan des réalisations, je pense que je dois progresser dans le domaine de l'Interface Homme-Machine (IHM) pour concevoir des outils plus ergonomiques.

6.4.2. Sur le plan humain

J'ai retiré un grand plaisir de la réalisation de ce mémoire, et je dois avouer que le stage d'un an en laboratoire constituait ma principale motivation pour le cursus CNAM. Et j'avais effectivement bien anticipé car j'ai bénéficié de l'ambiance d'une équipe soudée et chaleureuse, et ainsi que de l'ouverture internationale importante.

J'ai eu beaucoup de plaisir à travailler avec les gens du laboratoire, ceci malgré quelques difficultés personnelles. De mon côtoiement avec des personnes venues de tous les horizons, je retiens surtout ma découverte de la culture roumaine, dont le point fort est le voyage en Roumanie de 3 semaines avec Dia Miron.

La collaboration avec Raphaël Thomas et Serge Guelton fut fructueuse, et nos relations amicales perdurent. La rencontre des géographes, que ce soit l'équipe de Géophile à Lyon ou celle de Géographie-Cités de Paris a renforcé mon intérêt pour les sujets orientés sciences humaines, moi qui aimait déjà lire les magasines d'actualité sociale et économique comme « Courrier international », « Le nouvel observateur ».

En conclusion, ce fut un stage particulièrement enrichissant, sur tous les plans.

Ressources bibliographiques

[Andrienko, 1999] Andrienko G., Andrienko N., "Interactive Maps for Visual Data Exploration", *International Journal of Geographical Information Science, 13, p. 355-374*, 1999.

[Bertin, 1974] Bertin J., *Sémiologie Graphique*, Mouton, Gauthier-Villars, Paris, 2e édition, 1974

[Bissler, 2004] Bissler T., « Conception et développement d'une plate-forme pour la génération de Systèmes d'Information Spatio-Temporelle et Multimédia dédiés aux Risques Naturels », Mémoire d'ingénieur CNAM soutenu le 6 Juillet 2004.

[Bizet, 1997] Bizet F., « Carroyage et SIG urbain : les chômeurs à Rouen », *Mappemonde Vol2/97*, 1997

[Bloch, 2002] Bloch J, *Java Efficace, guide de programmation*, Vuibert, 2002. Traduction d'Alexis Moussine-Pouchkine.

[Booch et al., 2000] Booch G., Rumbaugh J., Jacobson I., *Le guide de l'utilisateur UML*, Eyrolles, Février 2000

[Boulier et al., 2003] Boulier J., Grasland C., « La forêt face au risque tempête : vers d'autres apports de l'analyse spatiale », *Colloque Théoquant* 2003, Besançon

[Brewer, 2004] Brewer C., « Color Use Guidelines for Mapping and Visualization », Chapter 7 (pp. 123-147) in *Visualization in Modern Cartography*, 1994.

[Bussi, 2000] Bussi M. , "*Les inégalités de santé en France : quels critères pour la répartition des resources publiques*" Actes du Festival International de Géographie de Saint Dié, 2000

[Cauvin, 1998] Cauvin C., "Des transformations cartographiques", *MappeMonde 49-1/98*, 1998

[Cavaness, 2002] Cavaness C., *Programming Jakarta Struts*, O'Reilly, 2002

[Chabert, 2007] Chabert C., « HyperAtlas et HyperAdmin : des outils cartographiques pour l'analyse de phénomènes sociaux », Mémoire d'ingénieur CNAM soutenu le 29 Mars 2007

[Charleux, 2003] Charleux L., « La politique régionale de l'union européenne : des régions à l'espace ? Essai d'analyse statistique et spatiale. » Thèse de doctorat. Université de Grenoble. 2003

[Cuenot, 2005] Cuenot O., « Modélisation spatiale multiscalaire de phénomènes sociaux », Mémoire d'ingénieur CNAM soutenu le 31 Mars 2005

[Danzart et al., 2003] Danzart A., Moissinac J.C., Potier C., « Standards pour la cartographie animée sur Internet ».-, Revue internationale de Géomatique, Volume 13 - n°1, 2003

[Denain et al., 1998] Denain J.-Ch, Langlois P., « Cartographie en anamorphose» *MappeMonde 49- 1/98*, 1998

[Dumas et al., 2001] Dumas E., Guérois M., « Une grille de lecture pour l'analyse des formes du peuplement en Europe - L'apport d'une méthode de lissage par

potentiels, in SIG et développement du territoire », *Revue Internationale de Géomatique, n°11, Vol3/4,* 2001

[Flanagan, 2002] Flanagan D., *Java in a Nutshell, 4ème édition, manuel de référence*, O'Reilly, 2002, traduction de Alexandre Gachet.

[François, 1996] François J-C, « Diffusion et dynamique des discontinuités: les élèves d'origine africaine dans les collèges de l'agglomération parisienne », *Mappemonde, n°4,* 1996

[Gotway et al., 2002] Gotway C., Young L., « Combining Incompatible Spatial Data », *Journal of the American Statistical Association, vol 97, n°458, pp 632-648,* 1 June 2002

[Gamma et al, 1999] Gamma E., Helm R., Johnson R., Vlissides J., *Design patterns. Catalogue de modèles de conception réutilisables,* Vuibert 1999. Traduction de Jean-Marie Lasvergères.

[Grasland et al., 2000] Grasland C., Mathian H., Vincent J.M., « Multiscalar Analysis and Map Generalization of Discrete Social Phenomena : Statistical Problems and Political Consequences.» *Statistical journal of the European Community,* 2000.

[Grasland, 2003] Grasland C., « Richesse et population dans le monde : une représentation multiscalaire des inégalités », *Mappemonde, n°69, p20-25,* 2003

[Grasland et al., 2003] Grasland C., Lizzi L., Martin H., Mathian H., Vincent J.M., « Hypercarte : un outil d'analyse spatiale multiscalaire des inégalités régionales en Europe ». *XXXIXème colloque de l'Association de Science Régionale de Langue Française,* Lyon, Septembre 2003.

[Grasland et al, 2005a] Grasland C., Guérin F., PACE, Terrier C., « La diffusion spatiale, sociale et temporelle des pièces euros étrangères : un problème complexe » *Actes des journées de Méthodologie Statistique*, 2005

[Grasland et al., 2005b] Grasland C., Martin H., Vincent J.M., Gensel J., Mathian H., Oulahal S., Cuenot O., Edi E., Lizzi L., « Le projet Hypercarte : analyse spatiale et cartographie interactive », *SAGEO*, 2005

[Grasland et al., 2006] Grasland C., Vincent J.M, *ESPON 3.4.3 : Final report for the MAUP. pp.151-168,* 2006

[Guérois, 2003] Guérois M., Les formes des villes vues du ciel. Contribution de Corine Land Cover à la comparaison des morphologies des grandes villes européennes, Thèse de doctorat, Université Paris 1, 2003

[Kraak et al., 2000] Kraak M-J., Brown A., *Web Cartography,* Taylor&Francis, London, 2000

[Lacaza et al., 2000] Lacaze M., Nirascou F., "Ces terres qui nous entourent…", *Les données de l'environnement, n°51, janvier 2000*

[Langlois et al., 1998] Langlois P., Lajoie G., « Cartographie par carroyage et précision spatiale » *Mappemonde Vol 49/1998.1*, 1998

[Larousse, 2002] *Le petit Larousse illustré*, Larousse, 2002

[Martin, 2004] Martin P., « Interface cartographique pour l'analyse territorialemultiscalaire de phénomènes sociaux », Mémoire d'ingénieur CNAM soutenu le 14 Décembre 2004.

[Neumann, 2003] Neumann A, Winter A., « La cartographie en mode vectoriel sur le Web : les possibilités de SVG ». Mars 2003, (accessible à http://www.carto.net/papers/svg/index_f.shtml)

[Nyquist, 1928] Nyquist H. « Certain Topics in Telegraph Transmission Theory », 1928 *Proceedings of the Institute of Electrical and Electronics Engineers, vol. 90, n°2, pp. 280-305*, réédition en 2002

[Plumejeaud, 2005] Plumejeaud C., « Quels outils et langages pour la cartographie libre sur le Web aujourd'hui ? » Probatoire du CNAM soutenu le 15 Décembre 2005.

[Plumejeaud et al, 2007] Plumejeaud C., Vincent J.M., Grasland C., Gensel J., Mathian H., Guelton S., Boulier J.,« Calcul et visualisation de cartes de potentiel interactives », SAGEO 2007.

[Poulain, 2004] Poulain M., Mario Pes G., Grasland C., Carru C., Ferrucci C., Baggio G., Franceschi C., Deiana L., "Identification of a geographic area characterized by extreme longevity in the Sardinia island: the AKEA study", *Experimental Gerontology, 39-9, pp. 1423-1429*, 2004

[Pumain et al., 2004] Pumain D., Saint Julien T., *L'analyse spatiale. 1, localisations dans l'espace*, Colin, 2004.

[Shepard, 1968] Shepard D. « A two-dimensional interpolation function for irregularly-spaces data. » *Proceedings of the 23rd National Conference*, New-York, ACM, pp 517-523.

[Tobler, 1979] Tobler W, "Smooth Pycnoplylatic Interpolation for Geographical Regions", *Journal of the American Statistical Association, vol. 74, n°367, pp. 519-536*, September 1979

[Tuckwell et al., 1998] Tuckwell H.C., Toubiana L., Vibert J.F, 1998. « Spatial epidemic network models with viral dynamics ». *Physical Review, 57:2*. 2163-2169.

Les sites Web suivants ont tous été consultés en janvier 2007.

[@AirNormand] http://www.airnormand.asso.fr/

[@Ajax] http://www.adaptivepath.com/publications/essays/archives/000385.php

http://www.scriptet.net/ajax-garrett.html

[@BetterSCM] http://better-scm.berlios.de/subversion/compelling_alternative.html

[@Brewer] http://www.personal.psu.edu/cab38/ColorBrewer/ColorBrewer.html

[@CommentCaMa http://www.commentcamarche.net/video/affich.php3

rche]

[@Elections2004-3D] http://www.princeton.edu/%7Ervdb/JAVA/election2004/

[@Elections2004-anamorphose] http://www.cscs.umich.edu/%7Ecrshalizi/election/

[@Europa] http://ec.europa.eu/eurostat/ramon/nuts/basicnuts_regions_fr.html

[@GeoClip] http://www.geoclip.fr/fr/p35_flashsvg.htm

[@GrandDico] http://www.granddictionnaire.com/btml/fra

[@HyperGeo] http://hypergeo.free.fr/article.php3?id_article=38

[@ICDG] http://www.geoconnections.org/publications/Technical_Manual/html_f/cgdiindex.html

[@INRIA] https://gforge.inria.fr/

[@LeMondeInformatique] http://www.lemondeinformatique.fr/actualites/lire-un-village-open-source-sur-le-salon-geo-evenement-2007-22515.html

[@NCGIA] http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/u40.html

[@Obfucateur] http://www.preemptive.com/obfuscation-faq/fr/index.html

[@Philcarto] http://philgeo.club.fr/Index.html

[@RennesAC] http://www.ac-rennes.fr/pedagogie/hist_geo/ResPeda/reseaux/cartes/isochroneligne.htm

[@SoftwareAG] http://www.softwareag.com/xml/about/glossary.htm

[@Wikipédia] http://fr.wikipedia.org/wiki/Client-serveur

http://fr.wikipedia.org/wiki/Service_Web

[@Web3D] http://www.web3d-fr.com/X3D/presentationx3d.php

[@W3C-SOAP] http://www.w3.org/2000/xp/Group/

[@W3C-SVG] http://www.w3.org/Graphics/SVG/

[@W3C-XMLHttpRequest] http://www.w3.org/TR/XMLHttpRequest/

ANNEXES

GLOSSAIRE

Analyse spatiale

L'analyse spatiale met en évidence des structures et des formes d'organisation spatiale récurrentes, et analyse des processus qui sont à l'origine de ces structures, à travers des concepts comme ceux de distance, d'interaction spatiale, de portée spatiale, …, de territorialité. Une attention particulière est apportée en analyse spatiale à la définition de l'échelon géographique considéré, du niveau d'observation, qu'il s'agisse du niveau " microscopique " des acteurs individuels ou d'agrégats spatiaux définis à des niveaux macro géographiques. L'analyse des processus par lesquels s'effectue le passage d'un niveau à un autre, l'émergence qualitative des structures qui rend pertinent le changement d'échelle, est l'un des grands problèmes posés actuellement à la réflexion théorique et à la modélisation en géographie. [@HyperGeo]

Architecture Client / Serveur

En informatique, système de coopération logicielle où l'une des parties (le serveur, acteur passif qui détient généralement les données) fournit des réponses aux requêtes émises par la seconde partie (le client, acteur actif, qui traite pour l'utilisateur final ces données). Les deux parties partagent le même protocole de communication, et le serveur est généralement capable de répondre à plusieurs clients simultanément. [@Wikipédia]

Carte

Représentation conventionnelle, généralement plane, de la répartition dans l'espace de phénomènes concrets ou abstraits. [Larousse, 2002]

Centroïde

Point fictif situé à l'intérieur d'un polygone convexe, dont les coordonnées correspondent approximativement à celles du centre de ce polygone.

Contiguité

Deux unités géographiques sont contiguës si elles ont une frontière en commun. Une matrice de contiguïté est un tableau carré dont les cases indiquent (en général par un ou zéro) si les unités géographiques portées en ligne et en colonne sont contiguës ou non. [@HyperGeo] Plus généralement, on peut considérer deux entités géographiques contiguës si elles se touchent, par une frontière, ou bien sont connectées par une ligne, une voie de communication. [Pumain et al., 2004]

Définition (d'une image)

Nombre de points (pixel) constituant l'image, c'est-à-dire sa « dimension informatique » (le nombre de colonnes de l'image que multiplie son nombre de lignes). Une image possédant 640 pixels en largeur et 480 en hauteur aura une définition de 640 pixels par 480, notée *640x480*. [@CommentCaMarche]

Discrétisation

La discrétisation d'une série statistique détermine des seuils numériques (qui seront associés à une certain trame ou couleur de visualisation) par la division de la série statistique en classes, en fonction du traitement voulu : quantiles, moyennes emboîtées, égales étendues, etc.
La discrétisation d'une étendue géographique consiste à diviser en parcelles régulières l'aire considérée, en projetant par exemple une grille dessus.

Généralisation

Opération consistant à générer un contour (l'enveloppe) géométrique correspondant à la forme d'une surface ou d'un volume réels (par exemple, le périmètre d'un bâtiment sur un cadastre). La généralisation se caractérise par son degré de finesse, c'est-à-dire du nombre de points utilisés pour décrire la forme.

Interactivité

En cartographie, l'adjectif s'applique si l'échelle et les informations affichées par la carte sont configurables et si les préférences sélectionnées de l'utilisateur s'appliquent immédiatement.

Internet

Réseau informatique mondial constitué d'un ensemble de réseaux nationaux, régionaux et privés, qui sont reliés par le protocole de communication TCP-IP et qui coopèrent dans le but d'offrir une interface unique à leurs utilisateurs. [@GrandDico]

Latitude

Mesure angulaire qui vaut 0° sur l'équateur, et s'étend jusqu'à 90° au pôle Nord, et -90° au pôle Sud. Tous les lieux d'un même parallèle ont la même latitude.

Longitude

Mesure angulaire qui vaut 0° sur un méridient de référence, et varie jusqu'à -180° en allant vers l'Ouest, ou +180° vers l'Est de ce méridien. On définit un méridien comme l'intersection du globe avec un plan orthogonal à celui de l'équateur, dont tous les points possèdent alors la même longitude. Le méridien de référence passe par Greenwich : sa longitude est 0°.

Multiscalaire

Relatif à l'observation sur des niveaux d'échelles différentes du même phénomène.

Mode Raster

Image construite à partir d'une matrice de pixels, auxquels on a affecté une couleur et une luminosité propre. La juxtaposition des pixels compose les formes des éléments : routes, fleuves, bordure. Les cartes topographiques numérisées, les photos satellites sont en général en mode raster.

Mode Vecteur

L'image se construit à partir de points de construction (deux points pour une ligne, un point et un rayon pour un cercle, etc.) auxquels sont affectées un rendu (couleur, style, transparence, etc.)

Obfuscation

Mécanisme de protection du code compilé, qui doit rendre l'analyse du code impossible tout en préservant son bon fonctionnement à l'exécution. En effet, avec un décompilateur de

Bytecode, comme *Java Decompiler* (JAD), un développeur peut aisément reconstituer le code source à partir d'un jar. [@Obfucateur]

Paquetage (*package*)

Collection nommée de classes (et éventuellement de sous-paquetages) permettant de grouper des classes apparentées et de définir un espace de désignation pour les classes qu'il contient. [Flanagan, 2002]

Projection

Technique de représentation de tout ou partie du globe terrestre sur une surface plane. La projection de Mercator en est un exemple. La carte est définie par son système de coordonnées (latitude/longitude) qui dépend de la projection utilisée.

Résolution (d'une image)

Souvent confondu avec la "définition", la résolution détermine par contre le nombre de points par unité de surface, exprimé en Points Par Pouce (PPP), (en anglais DPI pour *Dots Per Inch*); un pouce représentant 2.54 cm. La résolution permet ainsi d'établir le rapport entre le nombre de pixels d'une image et la taille réelle de sa représentation sur un support physique. [@CommentCaMarche]

Sérialisation

Opération permettant d'encoder un objet Java sous forme de flux d'octets, puis de le transférer d'une JVM à une autre, ou bien de le stocker de façon persistante pour une désérialisation ultérieure. La désérialisation étant l'opération inverse permettant de reconstruire un objet à partir de sa représentation sous forme d'octets. [Bloch, 2002]

Service Web (*Web Service*)

Un service Web est un ensemble de protocoles et de normes informatiques utilisés pour échanger des données entre les applications. [@Wikipédia]

INGENIERIE LOGICIELLE

Afin d'améliorer l'organisation du projet, et d'augmenter notre efficacité, nous avons enrichi le projet avec une gamme de nouveaux outils d'ingénierie logicielle. D'autre part, nous avons réalisé certains développements dont bénéficient tous les modules : mise en place d'un mécanisme de journalisation des évènements applicatifs, et la traduction des étiquettes et des messages de l'interface dans des fichiers séparés du code.

Méthodes de travail

L'ensemble de l'outillage proposé ici vise à aider l'ingénieur du projet dans les différentes tâches qui lui sont pour l'instant dévolues :

- le développement et la conception de nouveaux modules,
- la résolution des erreurs logicielles remontées par les utilisateurs,
- l'intégration de nouveaux jeux de données,
- et enfin la distribution et la diffusion des modules avec divers jeux de données.

a) L'environnement de développement

Dans un premier temps, nous mettons en place un environnement de développement adéquat pour une application comportant un volume très conséquent de code : pour information, il comporte 82719 lignes de codes, et 268 classes propres. Il est alors nécessaire d'utiliser un éditeur de code permettant une ré-ingenierie du code facile, si cela s'avère nécessaire, comme par exemple le re-nommage d'une méthode, ou bien le déplacement d'une variable dans une classe différente de sa classe d'origine. Ou encore, la possibilité de développer la pile d'appel d'une méthode permet de résoudre plus vite les erreurs. Nous utilisons pour cela Eclipse 3.1. En effet, Eclipse est diffusé largement et gratuitement et propose toutes les fonctionnalités citées. Nous ne citons pas toutes les fonctionnalités intéressantes (compilation, exécution et débuguer en ligne, intégration de ant, ou d'autres plugiciels (*plugins*) spécifiques, car Eclipse ne cesse d'évoluer. Nous aurions pu utiliser aussi XEmacs[38] qui propose aussi ce genre de fonctionnalités en tant que logiciel libre. Ces outils sont laissés à la préférence du développeur. Seulement, en raison des coûts de licence, nous laissons de côté les éditeurs commerciaux, comme JetBrains[39] qui propose pourtant avec IntelliJ IDEA 6.0 un produit intéressant et performant du point de vue ré-ingénierie du code, ou bien JBuilder de Borland[40].

Ainsi, le développeur a le choix d'un éditeur de code. Cependant il faut uniformiser les procédures de compilation et packaging du code : en effet, même compilés sur différentes machines, avec des environnements qui varient en fonction des développeurs, il est nécessaire que les fichiers binaires produits contiennent les même éléments nécessaires à une version, listés de façon formelle :

- les bibliothèques externes,
- les fichiers de propriété de l'application,
- le code compilé sur un sous-ensemble de classes défini pour chacun des modules,
- les données géographiques,

[38] http://www.xemacs.org/
[39] http://www.jetbrains.com/
[40] http://www.borland.com/fr/products/jbuilder/

- etc.

Les éditeurs de code offrent la possibilité de compiler le code, et de préparer dans des fichiers de configuration **au format spécifique** différentes distributions possibles. Mais ces fichiers justement ne sont pas compatibles entre éditeurs. Donc il nous est apparu nécessaire d'utiliser un outil pour la compilation et la préparation des distributions qui soit indépendant de l'éditeur de code : Ant[41]. Ant est un logiciel libre édité par la fondation Apache, et il a l'avantage d'être intégré dans éclipse via un *plugin*. Il s'utilise aussi en ligne de commande directement. Ant propose d'interpréter un fichier décrivant les taches de compilation et de distribution dans un format textuel, avec l'emploi d'une syntaxe XML.

Par exemple, dans notre fichier *build.xml*, interprété par ant, nous listons la liste des bibliothèques de fonctions (sous formes de fichiers d'extension .jar en java) qui seront nécessaires dans la distribution selon le module. Les phrases entre < !-- et --> sont des commentaires pour rendre le fichier lisible et compréhensible.

```
<!-- Unjar all external lib to prepare the full distribution
(avoid using complex classpath) -->
<target name="prepareDistHyperAdmin">
    <!-- for HyperAtlas  -->
    <unzip src="${lib}/jdom.jar" dest="${tmp}"/>
    <unzip src="${lib}/jts-1.5.jar" dest="${tmp}"/>
    <unzip src="${dist}/JGoodies.jar" dest="${tmp}"/>
    <unzip src="${lib}/log4j-1.2.14.jar" dest="${tmp}"/>

    <!-- for HyperAdmin  -->
    <unzip src="${lib}/deegree2004-02-13.jar" dest="${tmp}"/>
    <unzip src="${lib}/jxl.jar" dest="${tmp}"/>
    <unzip src="${lib}/postgis_1_0_0.jar" dest="${tmp}"/>
    <unzip src="${lib}/postgresql-8.1-404.jdbc3.jar" dest="${tmp}"/>
    <unzip src="${lib}/Field.jar" dest="${tmp}"/>

</target>
```

Figure 92. Extrait du build.xml : liste des bibliothèques nécessaires à la compilation de HyperAdmin

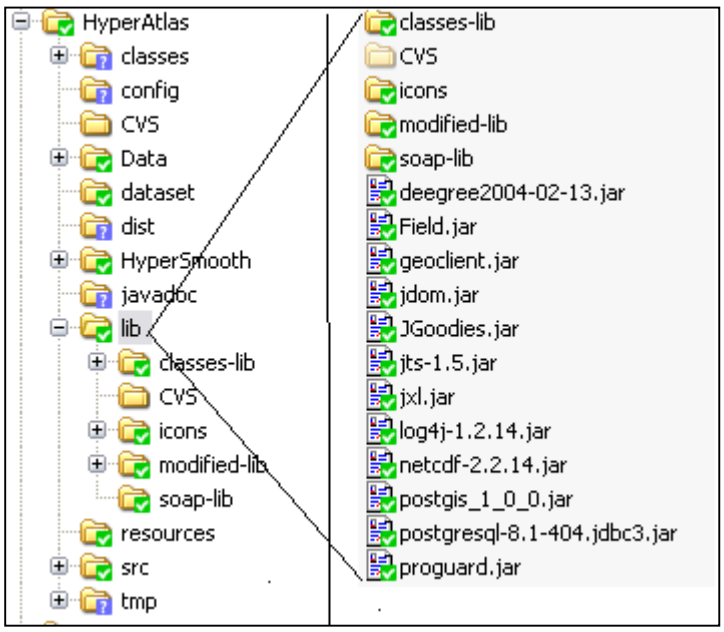
L'extrait de code précédent (cf. figure 92) montre la cible (*target*) qui est utilisée par ant pour compiler et distribuer *HyperAdmin*. *HyperAdmin* utilise, suite au travail de Christophe Chabert [Chabert, 2007], des bibliothèques de fonctions spatiales (postgis_1_0_0.jar et deegree2004-02-13.jar), des drivers d'accès à la base de donnée PostgreSQL (postgresql-8.1-404.jdbc3.jar), un parseur Excel (jxl.jar), en plus des jar utilisés par *HyperAtlas*.

Les sources du logiciel sont aussi ré-organisées: les fichiers que nous récupérons mélangent dans la même arborescence les binaires (.class), les classes Java (.java), les sources de la bibliothèque JGoodies et les images (ou icônes) employées dans l'interface. On ne connaît pas les versions de bibliothèques employées car elles sont simplement conservées dans un répertoire */lib*, sans leur numéro de version. Les fichiers de propriété et de licence sont perdus au milieu de l'arborescence du code. Nous décidons :

- de séparer le code source des classes compilées,
- d'utiliser un répertoire *dist/* pour envoyer le résultat d'une compilation,

[41] http://ant.apache.org/

- de renommer les bibliothèques avec leur numéro de version, stockées dans un répertoire spécifique *lib/*,
- de séparer le code modifié de JGoodies du code de *HyperAtlas*, et de placer dans un répertoire *lib/modified-lib/*, de manière à pouvoir recompiler la bibliothèque indépendamment du code principale et de préparer un jar JGoodies.jar chaque fois que nécessaire,
- de déplacer à la racine de l'arborescence les fichiers de propriété et de licence,
- de conserver toutes les images dans un répertoire *lib/icons/* spécifique.



La figure 93 est un aperçu de la structuration actuelle des sources, issue de l'évolution de cette organisation mise en place dès Février 2006.

Figure 93. Organisation du code en Janvier 2007.

b) La gestion de configuration

Enfin, un outil pour la gestion de configuration s'avère indispensable. Nous nous sommes d'abord intéressés à Libresource[42], une plateforme de travail collaboratif libre, qui propose des modules de synchronisation intégrés dans Eclipse. Cependant, malgré l'aide de Manuele Kirsch, thésarde spécialiste des environnements collaboratifs, nous avons mis plus de trois jours à mettre en place le serveur sur une machine de l'équipe, à cause d'un manque de maturité des procédures d'installation[43]. D'autre part, il s'est avéré qu'il nous était impossible pour des raisons de sécurité d'ouvrir une porte d'accès au serveur vers l'extérieur. Or l'équipe MESCAL qui doit pouvoir accéder aux sources pour contribuer sur la partie *HyperSmooth*, n'est pas située dans le même bâtiment, ni sur le même réseau.

Il s'avère que l'équipe MESCAL, à travers notre collaboration avec Saïd Oulahal, ingénieur de l'INRIA, nous ouvre le droit à l'ouverture d'un compte sur InriaGforge[44], un service pour faciliter les collaborations scientifiques des personnes travaillant à l'Inria[45], offrant « un accès aisé au meilleur de CVS (et de subversion), des listes de diffusion, de la gestion de bugs, de forums de discussion, de la gestion de tâches, de l'hébergement de sites, de l'archivage permanent de fichiers, de sauvegardes complètes » par le Web [@INRIA].

[42] http://dev.libresource.org/
[43] Attention, l'opération décrite s'est déroulée fin Février 2006.
[44] https://gforge.inria.fr/
[45] http://www.inria.fr/

InriaGforge impose de choisir pour le **gestionnaire de version** de code entre *Concurrent Versions System* (CVS) ou bien Subversion SCM (*Source Content Management*). Bien que Subversion comble certaines lacunes de CVS [@BetterSCM], nous avons choisi d'utiliser **CVS** car il était déjà connu des différents développeurs du projet, Christophe Chabert, Saïd Oulahal, et Christine Plumejeaud. Ce choix correspond à l'objectif de capitalisation du savoir-faire pour un maximum d'efficacité.

InriaGforge ne se limite pas à la gestion de version, il offre aussi les moyens d'assurer **le suivi des tâches et des bugs**. Nous donnons en annexe la liste des bugs recensés, et leur statut actuel (14 Février 2007) depuis l'ouverture de projet HyperCarte sur la forge. Le seul inconvénient est que pour enregistrer une anomalie dans le système, il faut avoir un compte donnant accès au projet HyperCarte sur la forge. En effet, notre projet n'est pas libre, et donc nous protégeons l'accès aux sources et à l'interface du projet : il est déclaré de statut privé. Nous avons donc créé un compte pour chaque membre du projet, avec un login et un mot de passe. Mais il s'avère que ceux-ci ont souvent perdu celui-ci. Et que la plupart du temps, le développeur a enregistré lui-même les erreurs qu'on lui remontait via d'autres canal de communication. L'avantage reste qu'il existe un endroit commun où les développeurs peuvent suivre l'avancée de leurs travaux.

Nous gérons aussi une liste de diffusion avec la forge permettant de diffuser des nouvelles à tous les membres du projet, et nous stockons les comptes-rendus de réunion sur cet environnement (cf. figure 94) ainsi que les documents de spécification, et les manuels des modules.

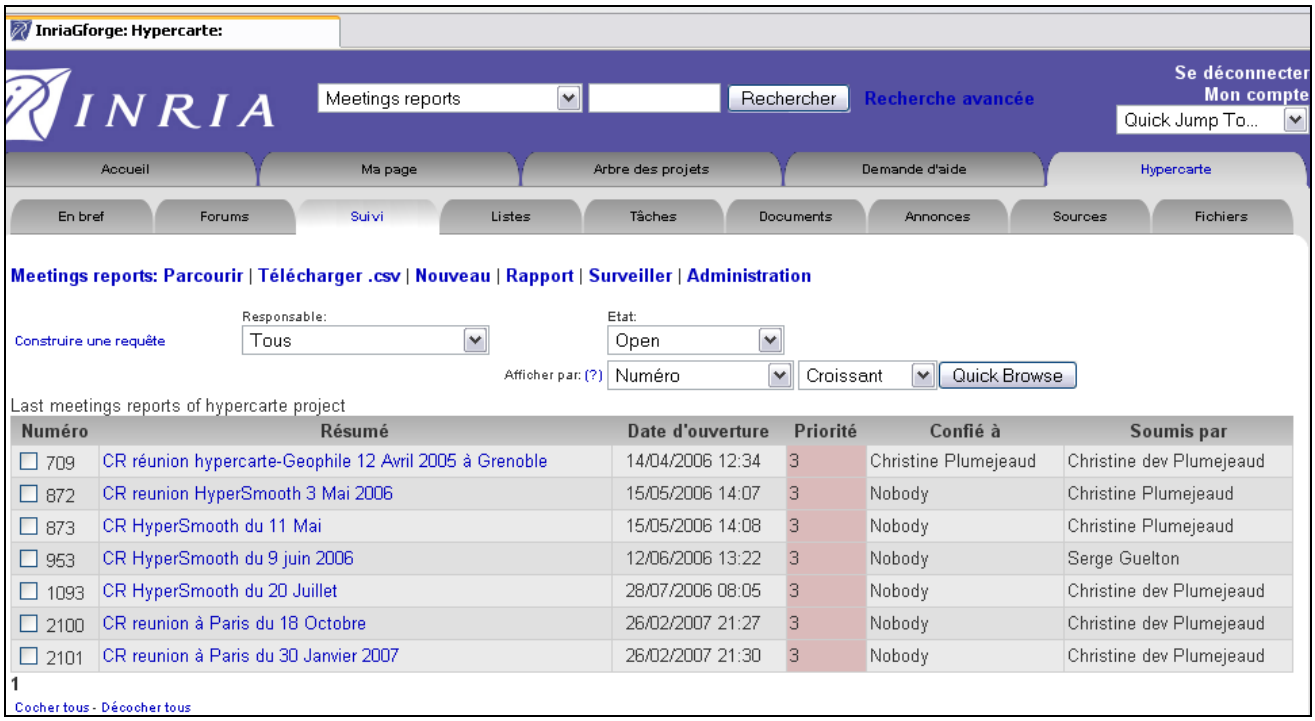


Figure 94. Interface de InriaGforge - accès aux comptes-rendus de réunion du projet Hypercarte.

Lors de la diffusion des versions, il manquait deux fichiers que l'on retrouve dans presque toute distribution de logiciel : **des fichiers README.txt et VERSIONS.txt**. Le

premier indique comment lancer l'application, et indique par exemple qu'elle fonctionne sous Java1.5. En effet, beaucoup d'utilisateurs utilisaient encore Java1.4 lors de la diffusion du logiciel en Mars, et les membres du projet, habitués à utiliser l'application sous Java1.4 étaient assez perturbés par ce changement. Le fichier VERSIONS.txt donne lui une liste des dernières versions diffusées, avec les améliorations et les fonctionnalités ajoutées, ainsi que le numéro de version actuelle.

c) La protection par obfuscation

Enfin, nous avons déjà fait remarqué que le logiciel n'est pas libre : nous déposons les sources à l'APP, et nous ne désirons pas laisser d'accès public sur les sources dans InriaGforge. Or il est possible pour un curieux d'accéder tout de même aux sources par un moyen assez simple : la compilation Java produit du bytecode, un langage interprété, qui « incluent les identifiants et les algorithmes des fonctions de l'application » [@Obfucateur]. L'utilisation d'un décompileur, comme *JAva Decompiler* (JAD) permet de reconstituer le code source, sans les commentaires, mais assez complètement. Pour protéger aux maximum les sources, il est donc nécessaire de pratiquer une opération nommée « **obfuscation** » pour brouiller le bytecode du jar distribué. Cette opération a l'avantage d'augmenter potentiellement les performances du code en diminuant la taille du programme exécuté par Java. Nous avons mené une recherche rapide pour trouver un obfuscateur adapté à nos besoins. ProGuard[46] semble répondre à nos besoins : c'est un logiciel sous licence GNU General Public Licence (GPL), reconnu, qui propose en plus un plugin intégré sous Eclipse. Il fonctionne aussi en ligne de commande, et il propose des directives en syntaxe XML adaptée pour s'exécuter directement à partir de Ant.

Il faut cependant faire attention au fait que *HyperAtlas* charge des fichiers de données sérialisées et en mars 2006, ces objets sérialisés ne possédaient pas d'identifiant unique de flux, connu sous le nom de numéro de série (*serialVersionUID*), qui est un champ permettant à Java de reconnaître le type de l'objet à désérialiser [Bloch, 2002]. Pour supporter le brouillage du code, un objet sérialisé doit comporter ce champ. En effet, lorsqu'il est absent, Java le calcul, en se basant sur les membres de la classe et les signatures des méthodes. Or l'obfuscation modifie justement ces éléments, et empêche donc à l'exécution de reconnaître le type des objets sérialisés. Tant que le modèle de données ne comportait pas un champ *serialVersionUID* pour chaque type d'objet, il était impossible d'ouvrir dans un *HyperAtlas* obfusqué un jeu de données sérialisées. Aujourd'hui, suite aux évolutions de HyperAdmin, ce problème est résolu.

Le second problème rencontré est que nous devons alors aussi obfusquer JGoodies, une bibliothèque graphique qu'utilise *HyperAtlas* suite à la contribution de Olivier Cuenot [Cuenot, 2005]. En effet, l'application n'utilise pas d'inclusion de bibliothèques externes à l'exécution via l'usage de la directive *–classpath*. Elles sont incluses directement dans le jar constituant la distribution. Or suite à diverses manipulations, un des fichiers sources de notre adaptation de JGoodies[47] était manquant (qui depuis est devenue un produit commercial, l'accès aux sources n'est plus libre). Ceci empêchait la recompilation de JGoodies, et par la même son obfuscation. Ultérieurement, nous avons trouvé le fichier manquant, mais par manque de temps, nous n'avons pas re-testé la procédure d'obfuscation. Cependant le travail

[46] http://proguard.sourceforge.net/
[47] http://www.jgoodies.com/

est préparé : une cible adaptée est écrite dans le fichier build.xml, et proguard.jar est stocké dans CVS.

Développements communs

Nous relatons ici deux améliorations notables apportées à l'ensemble du code de *HyperAtlas*, et qui se sont reportées naturellement dans les modules *HyperAdmin* et *HyperSmooth*. A la suite des demandes d'améliorations d'ESPON, nous avons mené une analyse du code, permettant de dégager la structure du logiciel. Grâce à cette analyse, nous pouvons aisément apporter les améliorations requises par le contrat et imaginer comment connecter les nouveaux modules aux fonctionnalités existantes, afin d'exploiter le code de *HyperAtlas*, sans le dupliquer.

a) Internationalisation des messages applicatifs

Le contrat d'ESPON demande par exemple le changement de certains libellés dans l'interface graphique. A cette occasion, nous nous rendons compte que tous les libellés sont dispersés dans les diverses classes du code, en anglais, avec parfois des fautes d'orthographe. Pour fabriquer une distribution traduite en français ou en roumain, il faudrait dupliquer le code. De plus, la maintenance des messages applicatifs est plutôt inconfortable. Notre expérience des technologies Web va nous conduire à mettre en place un système de centralisation des étiquettes et messages, avec traduction dans la langue de l'utilisateur. C'est **l'internationalisation** de l'application.

La centralisation des messages se fait sous forme d'un fichier texte contenant des clés, symbolisant toutes les étiquettes et messages de l'application, avec leur traduction dans une langue. La syntaxe d'une entrée dans le fichier est <clé>=<traduction>. Il existe un fichier par langue :

- français dans le fichier HyperCarteResources_**fr**.property,
- anglais dans le fichier HyperCarteResources_**en**.property,
- roumain dans le fichier HyperCarteResources_**ro**.property,
- etc.

Ces fichiers sont des ressources pour l'application, que nous rangeons donc dans un répertoire dédié sous la racine du projet : *resources/.*

La figure 95 donne un exemple de contenu pour le fichier de traduction anglaise, et nous avons surligné en gras les clés, et en gris les commentaires. C'est un fichier de propriétés qui se lit au démarrage de l'application par un mécanisme simple de chargement des propriétés existant en Java. ESPON demandait par exemple de mettre à jour l'étiquette **`espon.border.notice`** et **`espon.border.copyright`** dont le texte est une mention apposée sur le côté droit des cartes de l'atlas.

```
#HyperCarte messages (english)
message.error=HyperAtlas Error
message.info=HyperAtlas Message
espon.label.loading=ESPON HyperAtlas is loading ...
label.loading=HyperAtlas is loading ...
applicationName=HyperAtlas - Multiscalar Territorial Analysis
espon.applicationName=ESPON - HyperAtlas - Multiscalar Territorial
Analysis
message.java.version=HyperAtlas need Java version 1.5 or higher to run
(1.5 is recommanded)\n You can download it at \"java.sun.com\"
message.plafTheme.error=Choosen plafTheme wasn''t found !\n Replacing by
default plafTheme.
message.scale.absent=No scale available
#General tooltip
tooltip.minimize=Minimize
tooltip.maximize=Maximize
tooltip.restore=Restore
tooltip.zoom=Zoom
#Titles of maps
title.map.context=Study Area and Elementary Zoning
title.map.globalDeviation.value=Deviation to Value {0}
title.map.globalDeviation=Deviation to {0}
title.map.mediumDeviation=Deviation to {0}
title.map.localDeviation=Deviation to {0}
title.map.synthesis=Synthesis of Deviations
#Short description of maps (used for tab names in MaintabbedPane)
shortDesc.map.context=Area and Zoning
shortDesc.map.numerator=Numerator
shortDesc.map.denominator=Denominator
shortDesc.map.indicator=Ratio
shortDesc.map.globalDeviation=Global Deviation
shortDesc.map.mediumDeviation=Medium Deviation
shortDesc.map.localDeviation=Local Deviation
shortDesc.map.synthesis=Synthesis
#ESPON border text
espon.border.notice=The maps and figures involved in this report are
realised under the responsibility of the user and do not necessarily
reflect the opinion of the ESPON Monitoring Committee.\n
espon.border.copyright=\u00a9 Eurogeographics association, MCRIT and
RIATE for administrative boundaries\n\nOrigin of data:\n- ESPON
database, January 2006\n- World Developpement Indicators 2003
…
```

Figure 95. Début du fichier de traduction des messages en anglais.

La mise en place de ce mécanisme nécessitait donc de parcourir tout le code pour trouver des messages applicatifs, leur créer une clé dans le fichier de ressource et donner la traduction. Nous utilisons un objet dédié au chargement des propriétés et à la traduction des messages, *HCResourceBundle*, qui est un singleton (instance statique publique unique). Ce schéma singleton est justifié puisqu'il agit comme une fabrique de messages pour l'ensemble de l'application [Gamma, 1999]. Dans le cas où nous n'avons pas de fichier de traduction dans la langue utilisateur, nous utilisons l'anglais comme traduction par défaut. Par ailleurs, si une clé n'a pas sa traduction dans le fichier, nous affichons alors dans l'application la clé elle-même.

Le constructeur prend comme paramètres le nom du fichier de ressource et la langue de l'utilisateur, avec lesquels il détermine le fichier dans lequel il doit récupérer les traductions, qui se présentent sous la forme d'une liste de propriétés (<clé> = <valeur>). Lorsque l'on souhaite connaître la traduction d'un message, on accède à ces propriétés par la clé.

Ces messages peuvent être adaptés avec un contenu dynamique : pour cela, nous utilisons les fonctions de formatage des chaînes de caractères fournies dans le paquetage *java.text.MessageFormat* : les parties numérotées *i* entre accolades du message seront remplacées par l'objet d'indice *i* du tableau passé en paramètre dans la méthode. Nous donnons le code de cette méthode en figure 96.

```
/** Exemple : message.info=HyperAtlas show {0} maps, analyse done at {1}
 * args[0] = 8
 * args[1] = 12 March 2006
 * this format("message.info", args) will return : HyperAtlas show 8 maps,
 * analyse done at 12 March 2006
 * @param key : key of message to be translated, this message string may
 *       contain some data to be replaced
 * @param args : array of values used for replacement in message string
 * @return the message key translated, with values completed
 */
public String format(String key, Object[] args){
    MessageFormat msgFormat = new MessageFormat((String) handleGetObject(key));
    msgFormat.setLocale(_currentLocale);
    return msgFormat.format(args);
}
```

Figure 96. Code extrait de *HCResourceBundle* : *format*().

L'exemple suivant (cf. figure 97) montre comment nous utilisons cette fonctionnalité sur la titre de la carte de déviation globale : soit la déviation globale est rapportée à la moyenne des ratios sur l'aire d'étude, soit elle se rapporte à une valeur éditée par l'utilisateur. C'est un cas où l'adaptation du libellé s'avère nécessaire. En commentaire, l'ancienne méthode d'écriture des libellés.

```
case Settings.MAP_GLOBAL_DEVIATION:

    if (Settings.getInstance().isGlobalDeviationValueSelected()) {
        Object[] args  = {Settings.getInstance().getGlobalDeviationValue()};
        title = Settings.getInstance().getResourceBundle()
            .format("title.map.globalDeviation.value", args);
        //title = "Deviation to Value "+ Settings.getInstance().getGlobalDeviationVa
    } else {
        Object[] args  = {Settings.getInstance().getGlobalDeviationName()};
        title = Settings.getInstance().getResourceBundle()
            .format("title.map.globalDeviation", args);
        //title = "Deviation to "   + Settings.getInstance().getGlobalDeviationName
    }
    break;
```

Figure 97. Utilisation des traductions avec contenu adaptable.

Ce mécanisme d'internationalisation profite aux trois modules, qui ont chacun leur fichier de ressources : HyperCarteResources*.property, HyperAdminResources*.property, HyperSmoothResources*.property.

b) Suivi des évènements applicatifs dans un journal de bord (logging)

Lors de nos premiers développements, afin d'apprendre à mieux connaître l'application, nous utilisons une manière très simple de connaître les actions internes du logiciel : nous insérons des instructions de sortie sur notre terminal, *System.out.println()*, dans les méthodes qui nous semblent intéressantes. Ainsi, les traces applicatives défilent dans notre terminal et nous renseignent sur l'ordre des opérations, les valeurs de telles ou telles variables internes de l'application.

Cependant cette façon de loguer est contre performante pour le système : cela ralentit les calculs. D'autre part, les logs ainsi produits ne sont pas sauvés dans un fichier texte, et si par exemple ils doivent servir à mesurer les performances du logiciel ou bien à récupérer le contexte détaillé d'une erreur, il faudra alors « bricoler » des copier-coller depuis le terminal. De plus, lorsque nous fabriquons une distribution pour les utilisateurs finaux, nous devons mettre en commentaire ces instructions dans le code. C'est un travail supplémentaire fastidieux, et ces commentaires gênent la lisibilité du code.

Nous voudrions utiliser un mécanisme nous permettant :

- de changer le niveau des sorties (pour un journal plus ou moins verbeux),
- de sélectionner le type de sorties désiré : fichier, terminal ou bien tampon mémoire,
- de formater automatiquement les messages avec la date, le nom de la classe et de la méthode qui logue,
- de moduler ces paramètres en fonction du paquetage de code, ou bien du nom des classes.

Il existe un paquetage standard *java.util.logging* autorisant toutes ces opérations fourni avec le kit de développement Java 1.4. Nous lui préférons cependant un paquetage fourni par la fondation apache et très largement utilisé dans tous ses produits dérivés (Jakarta Struts par exemple) : *org.apache.log4j*. Log4j[48] est simple d'utilisation et bien documenté ; il est conçu pour que les traces n'altèrent pas les performances de l'application. Les traces peuvent être activées en cours d'exécution, et pour changer le niveau de log, aucune recompilation n'est nécessaire.

Notre contribution consiste à créer une fabrique de logs, *HCLoggerFactory*, basée sur l'utilisation du paquetage *org.apache.log4j*, et de fournir un fichier de configuration de niveau de log à l'application. La quantité et la criticité des informations enregistrées sont proportionnelles au niveau des logs. Ainsi, le log peut-être utilisé par le développeur pour déboguer (DEBUG), et dans ce cas, on préfèrera ne pas voir ces traces en production ; le log peut aussi être informatif (INFO), ou bien donner des avertissements (WARN), ou encore décrire des erreurs survenues dans l'exécution (ERROR). Les niveaux hiérarchiques dans l'ordre croissant sont : ERROR, WARN, INFO, DEBUG. En loguant au niveau *n*, les messages de niveau *n+1* sont ignorés.

[48] http://logging.apache.org/log4j/docs/index.html

Donc le code de *HCLoggerFactory* est extrêmement simple : il respecte le modèle singleton [Gamma, 1999], et précise le nom et l'emplacement du fichier de configuration : **logger.property.** Ce fichier doit se trouver dans le même répertoire que le jar applicatif au moment de l'exécution, sans quoi l'application est privée de journal de bord (mais elle fonctionne).

Notre configuration *logger.property* est conçue pour :

- sortir sur la console tous les messages de niveau ERROR
- créer un journal de bord par exécution, et y accumuler les logs, suivant les niveaux spécifiés par module. Pour une distribution aux utilisateurs, il vaut mieux préciser un niveau INFO pour tous les modules.

De plus les messages sont formatés pour indiquer la date et l'heure de log, ainsi que le nom de la classe qui produit la trace.

La sortie est définie comme un « *appender* » dans le fichier de configuration. Et l'objet de la figure 98 est de montrer comment nous configurons dans *l'appender* A2 une sortie vers un fichier nommé « logs.txt » dans le répertoire personnel de l'utilisateur, avec le type de mise en forme :

```
# Appender A2 writes to the file "logs" in user's home.
log4j.appender.A2=org.apache.log4j.FileAppender
log4j.appender.A2.File=${user.home}/Mes\ documents/temp/logs.txt
# Truncate 'logs.txt' if it aleady exists.
log4j.appender.A2.Append=false
# Appender A2 uses the PatternLayout.
log4j.appender.A2.layout=org.apache.log4j.PatternLayout
log4j.appender.A2.layout.ConversionPattern=%d{dd/MM/yyyy  HH:mm:ss}  %-5p
[%c{2}] %m%n
```

Figure 98. Premier extrait du fichier de configuration des logs (logger.property).

Ensuite, la configuration repose sur un système arborescent, où on peut définir un certain niveau de log pour la racine, (le plus haut paquetage), puis raffiner le niveau désiré en fonction des sous-paquetages ou bien même des classes. Ce sont les niveaux définis au niveau des feuilles qui prévalent. Avec la figure 99, nous montrons un exemple de configuration du fichier pour utiliser HyperAdmin en mode de développement : les traces de *HyperSmooth* sont au niveau ERROR, celles de HyperAdmin au niveau DEBUG, tandis que certaines classes clés de *HyperAtlas* ont un niveau de log plus verbeux.

```
log4j.logger.hypercarte.hyperadmin=DEBUG, A2
log4j.logger.hypercarte.hypersmooth=ERROR, A2

log4j.logger.hypercarte.hyperatlas.config.Settings=DEBUG, A2
log4j.logger.hypercarte.hyperatlas.io.HCSerialInputQueries=DEBUG, A2
log4j.logger.hypercarte.hyperatlas.misc.HCUnitRepository=DEBUG, A2
log4j.logger.hypercarte.hyperatlas.serials.DataSerialver=DEBUG, A2
log4j.logger.hypercarte.hyperatlas.serials.SerialElement=INFO, A2
log4j.logger.hypercarte.hyperatlas.ui.AbstractMap=DEBUG, A2
```

Figure 99. Second extrait du fichier de configuration des logs (logger.property).

Enfin, l'usage de notre fabrique est identique dans tous les objets de HyperCarte : ils demandent une instance de *logger*, en donnant en paramètre leur nom de classe. Souvent, l'instance de *logger* est enregistrée comme un attribut de classe, affecté lors du premier usage de l'objet, il est ensuite accédé aussi souvent que nécessaire.

c) Organisation des paquetages

Ces développements, internationalisation et *logging*, sont utiles et utilisables par les 3 modules : *HyperAtlas, HyperSmooth* et *HyperAdmin*. Nos prochains développements vont s'attacher à réutiliser le plus possible les classes de *HyperAtlas* mais sans surcharger le code de *HyperAtlas* avec les nouvelles fonctionnalités. Dans ce but, nous créons une arborescence de paquetage dont ***hypercarte*** est la racine.

Le code existant de *HyperAtlas* est restructuré dans un paquetage ***hypercarte.hyperatlas***. A l'intérieur de ces paquetages, nous regroupons :

- la partie graphique dans un paquetage ***ui***,
- la partie entrée-sorties sur le modèle de données dans un paquetage ***io,***
- la définition des évènements dans un paquetage ***event,***
- la configuration de l'application et le paramétrage des analyses dans un paquetage ***config***,
- les classes utilitaires (comme *HCLoggerFactory*) sont dans ***misc***.

L'intérêt de notre ré-organisation de code réside dans le fait que nous pouvons ainsi compiler séparément les modules : *HyperSmooth* a des dépendances sur *HyperAtlas*, mais l'inverse n'est pas vrai : *HyperAtlas* ne contient aucun code spécifique à *HyperSmooth*, ni à *HyperAdmin*.

Il nous reste un souci dans cette réorganisation : le paquetage ***data*** qui regroupe le modèle de données sérialisées ne peut pas être renommé ni déplacé. Du fait de l'absence du champ *serialVersionUID*, le problème déjà décrit précédemment pour l'obfuscation se pose : le paquetage ne supporte aucune modification, sous peine de ne plus pouvoir ouvrir les anciens jeux de données. Le nouveau de jeu de données sérialisées pour *HyperAtlas* que produit *HyperAdmin* est placé lui dans le paquetage ***hypercarte.hyperatlas.serials***.

METHODES D'INTERPOLATION SPATIALE

Cette annexe présente quelques méthodes numériques d'interpolation pour la représentation continue de phénomènes géographiques, à partir de données connues en quelques points géoréférencés. Nous ne discutons pas les méthodes qui se basent sur des données collectées par surface (comme la méthode pycnophylactique). L'interpolation est nécessaire puisqu'il est impossible de collecter de façon exhaustive les données en tous les points de l'espace, ceci pour des raisons pratiques évidentes (coûts, inaccessibilité…). Or l'interpolation est un moyen permettant de générer de l'information sur les points de l'espace non enquêtés, pour la cartographie et l'analyse en 2D du phénomène. Les SIG intègrent de plus en plus souvent des fonctionnalités avancées d'interpolation qui opèrent une discrétisation de l'espace géographique, (sous la forme d'une grille le plus souvent), et proposent aux utilisateur le choix de la méthode d'estimation.

Ces méthodes peuvent alors intégrer des hypothèses sur le phénomène analysé sur l'ensemble du domaine d'étude (on les qualifie alors de globales), ou se contenter d'appliquer répétitivement un algorithme sur des petites portions de l'aire (cas des méthodes dites locales). On distingue aussi les méthodes déterministes des méthodes stochastiques (ou géostatistiques) : ces dernières présupposent une distribution aléatoire du phénomène, et permettent d'estimer l'incertitude sur le résultat produit. On oppose aussi les méthodes dites exactes au méthodes d'approximation : dans le premier cas, la surface obtenue passe exactement par les points de mesure (cas du Krigeage, des fonctions B-splines par exemple). [@NCGIA]

Ici, nous classifions les méthodes d'interpolation suivant deux approches : l'approche déterministe et l'approche probabiliste.

L'approche déterministe regroupe toutes les méthodes d'interpolation dont la fonction de structure est choisie à priori, et qui ne fournissent aucun renseignement sur la fiabilité de l'estimation (sa variance). On veut connaître en tout point *M* de coordonnées *(x,y)* la valeur *F(x,y)*, sachant que l'on connaît les valeurs f_i en les points M_i, pour *i* variant de *1* à *n*. Dans cette catégorie, les méthodes les plus connues à ce jour sont la méthode polynomiale (ou surfaces de tendance), les fonctions B-splines, l'inverse de la distance, la triangulation, la méthode de Shepard.

- **Méthode polynomiale** (ou analyse en surfaces de tendance)

Cette méthode utilise la combinaison linéaire de fonctions polynomiales. Il s'agit d'appréhender le phénomène par l'utilisation d'un polynôme dont le degré est choisi par l'utilisateur. Ainsi, d'une manière générale, on écrit que

$Z = F(x,y)=a+b_1x+b_2y+b_3xy+b_4x^2+b_5y^2+\ldots$

Les paramètres du modèle (a, b_1, b_2, b_3, b_4, b_5, …) sont estimés par la méthode des moindres carrés qui vise à minimiser l'erreur entre la valeur prédite Z_i et la valeur observée f_i.en minimisant la somme [7]:

$$\sum_{i=1}^{n}(Z_i - f_i)^2 \qquad [7]$$

Cette méthode globale s'intéresse peu aux changements locaux, aux irrégularités, mais seulement à la tendance. Le choix du degré fixe en effet la complexité de l'interpolation et

dans la pratique on utilise des degrés de 3 à 5. Cette méthode d'interpolation à l'inconvénient de s'accroître ou de décroître rapidement dans les zones où l'échantillonnage est faible et particulièrement sur les bords de la zone.

- **Les fonctions B-splines**

Avec cette méthode locale, on interpole par morceaux la surface à l'aide de fonctions polynomiales, en contrôlant la forme de la fonction entre chaque point de mesure, et surtout son degré de continuité (classe C^0, C^1, C^2)[49]. La surface estimée recoupe chaque point de mesure. Cette méthode produit des surfaces très lissées (c'est-à-dire sans changement abrupt d'altitude), et convient donc très bien aux jeux de données présentant des variations lentes.

Pour les méthodes suivantes, l'idée commune est d'opérer une pondération sur les valeurs f_i, en supposant que le poids λ_i affecté à chaque f_i décroît proportionnellement avec sa distance au point *M*. Cette hypothèse est issue de la première loi de géographie émise par Tobler : les lieux proches ont plus de chance d'avoir des valeurs similaires que les lieux éloignés. On cherche donc à résoudre au mieux l'équation [8].

$$F(x,y) = \sum_{i=1}^{n} \lambda_i f_i \qquad [8]$$

On pose comme les conditions suivantes sur les poids λ_i :

- leur somme est égale à 1,
- ils sont toujours positifs ou nuls,
- ils dépendent de la distance entre l'échantillon *i* et le point estimé *M*.

- **Inverse de la distance**

Il s'agit ici d'affecter aux poids de pondération λ_i une valeur directement proportionnelle à l'inverse d'une distance *d* élevée à une puissance *n*, *d* étant la distance séparant le point estimé M des points d'observation, cf. l'équation [9].

$$\lambda_i = \frac{k}{d_i^n} \qquad [9]$$

Cette méthode permet d'obtenir des grilles très rapidement mais crée des zones circulaires autour des points observés (effet *Bull'eyes*). Cette aspect peut-être lissé en jouant sur la puissance et le voisinage. C'est un interpolateur exact (il passe par les valeurs observées).

- **Triangulation**

Elle s'appelle aussi méthode des **plus proches voisins**, ou encore des **polygones de Thiessen** ou de Voronoï ou encore de Dirichlet. C'est un cas particulier de la fonction inverse, avec *n=1* et tous les coefficients λ_i nuls exceptés pour les trois plus proches voisins de M : on utilise les trois points les plus proches du point à estimer. Elle correspond à un lissage par voisinage à trois point. L'espace autour des points de mesure est découpé en polygones de telle manière que chacun comprenne toutes les positions possibles pour lesquelles il est le plus proche point d'échantillonnage et on attribue la valeur du point à tout le polygone. Les limites de cette méthode sont les suivantes :

- elle s'applique mal aux variables évoluant de manière progressive,

[49] De classe C^0 une fonction est dite continue, de classe C^1 elle est dérivable, et de classe C^2, sa fonction dérivée seconde est continue, et dans ce cas, la surface obtenue présente une courbature minimale.

- elle entretient l'illusion de phénomènes inchangés entre les limites des polygones et qui changent de manière brusque à la frontière,
- elle dépend en fait totalement de la distribution des points d'échantillonnage.

- **Méthode de Shepard**

La méthode de Shepard [Shepard, 1968], et ses dérivées sont une adaptation de la méthode de la fonction inverse avec $n=2$ (les poids sont proportionnels à une puissance carrée inverse de la distance), visant à supprimer l'effet *Bull'eyes* en intégrant un critère de moindres carrés. En effet, on remplace les valeurs f_i par une meilleure approximation locale polynomiale quadratique $Q_i(x,y)$. En outre, on introduit la notion de portée r_w : au-delà d'un certain rayon à partir du point estimé M, les poids sont nuls. Ceci à pour effet de rendre la méthode locale, et surtout d'accélérer les calculs. Cette méthode est très largement répandue dans les SIG.

Ces différentes méthodes sont très faciles d'utilisation, et elles permettent d'avoir rapidement une idée des zones présentant de fortes valeurs et des zones avec des valeurs faibles. Cependant, elles présentent deux inconvénients majeurs :

– il est impossible de connaître la fiabilité de l'estimation.
– la connaissance de la structure spatiale du phénomène n'est pas mise à profit.

L'approche stochastique comble ces lacunes. Les principales méthodes connues à ce jour sont la méthode du Krigeage (et ses multiples dérivées : co-Krigeage, Krigeage glissant) et les méthodes d'analyse par série de Fourier.

- **Le krigeage** (ou meilleure estimation linéaire non biaisée)

Cette méthode est née avec l'exploitation minière en Afrique de Sud et porte le nom de son inventeur (D.G. Krige), puis a été développée en 1963 par Georges Matheron avec « la théorie des variables régionalisées ». Elle vise toujours à résoudre l'équation [8], cependant la règle de pondération et donc la carte qui en résulte sont directement déterminées par le comportement spatial des données. Pour quantifier la continuité spatiale du phénomène, on établit un variogramme : celui-ci mesure le degré de similarité entre les points en fonction de leur éloignement.

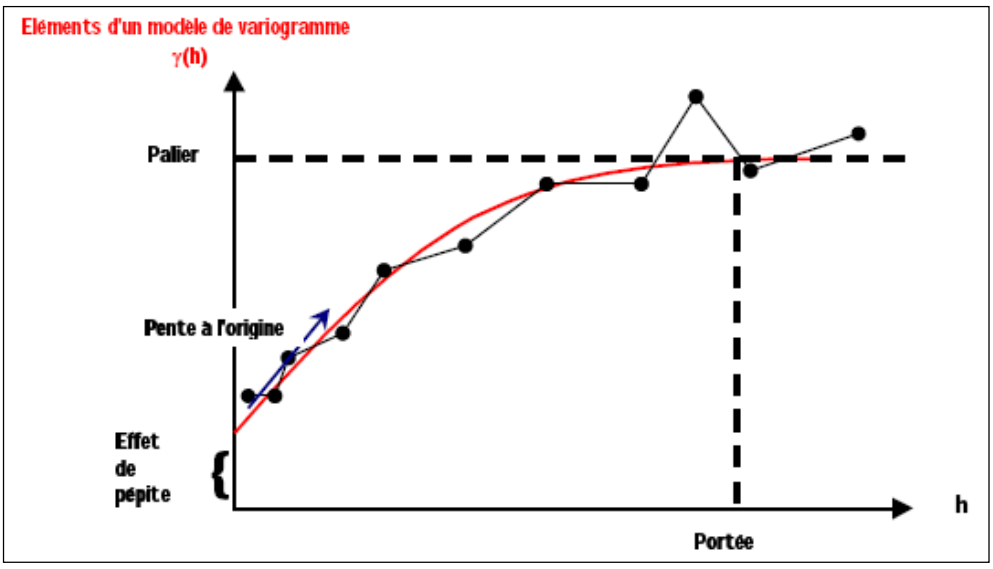


Figure 100. Exemple de variogramme.

Les éléments d'un modèle de variogramme sont les suivants (cf. figure 100) : l'axe des ordonnées porte l'espérance moyenne de la différence de mesure entre deux points séparés d'une distance *h*, que l'on porte sur l'axe des abscisses. On constate qu'au-delà d'un certaine distance (la portée), la variance se stabilise autour d'un certain pallier, et théoriquement, les points séparés par une distance supérieure à cette portée sont non corrélés. A l'origine du variogramme, on mesure le degré d'irrégularité du phénomène : avec une tangente verticale, le phénomène est très irrégulier, ou alors les données présentent beaucoup d'erreurs de mesure. Pour établir le variogramme, il faut choisir des classes de distances et de directions dans l'espace. Ce choix est délicat, il influence beaucoup les résultats de la méthode.

Le modèle du variogramme est ensuite intégré dans un système linéaire d'équation dit système de krigeage afin de déterminer la valeur optimale des poids λ_i affectés à chaque f_i. De cette façon un poids plus grand est affecté aux données qui sont mieux corrélées avec le phénomène au nœud considéré. On connaît aussi alors la variance (ou degré de fiabilité) associé à chaque nœud : elle représente la dispersion possible de la valeur réelle dans une fourchette centrée sur la valeur estimée. Plus cette variance est faible, plus la valeur estimée est proche de la réalité.

Le co-krigeage (ou krigeage avec dérive externe) permet lui en plus d'intégrer la connaissance sur la répartition d'une variable auxiliaire, que l'on suppose déterminante sur la diffusion du phénomène étudié (comme l'altitude dans les phénomènes de diffusion de polluants atmosphériques). Le système de krigeage est alors établi à partir de la corrélation croisée entre la variable principale et son auxiliaire. Dans le cadre d'études sur la diffusion des polluants atmosphériques, on observe que l'emploi d'une dérive externe améliore sensiblement la qualité des cartes obtenues [@AirNormand]

Cette méthode repose sur un certain nombres d'hypothèses qui doivent être vérifiées : elles concernent le type de distribution des fréquences des valeurs f_i ainsi que des conditions sur la non stationnarité de la variance. Les mathématiciens montrent que l'on peut limiter les conséquences négatives du non respect des hypothèses de non stationnarité en réalisant un krigeage glissant.

Avec le krigeage glissant, au lieu d'établir un énorme système de krigeage en considérant l'ensemble de tous les points du domaine d'étude, on se limite aux points contenus dans un cercle de rayon *r*, centré sur le point à estimer. La matrice ainsi obtenue est nettement plus petite (donc le système plus facile à résoudre), mais il faut réitérer la méthode sur chaque point à estimer, ce qui rend cette méthode est nettement plus longue et coûteuse en calculs que le krigeage simple.