CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

MEMOIRE

présenté par Philippe MARTIN

en vue d'obtenir

LE DIPLÔME d'INGENIEUR C.N.A.M.

en INFORMATIQUE

Interface cartographique pour l'analyse territoriale multiscalaire de phénomènes sociaux

Soutenu le 14 décembre 2004

JURY

Présidente : Mme Véronique Donzeau-Gouge

Membres : M. Eric Gressier
M. Jacques Courtin
M. Jean-Pierre Giraudin
M. Jérôme Gensel
Mme Paule-Annick Davoine

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

MEMOIRE

présenté par Philippe MARTIN

en vue d'obtenir

LE DIPLÔME d'INGENIEUR C.N.A.M.

en INFORMATIQUE

Interface cartographique pour l'analyse territoriale multiscalaire de phénomènes sociaux

Soutenu le 14 décembre 2004

Contexte général du mémoire présenté (ingénieur **C**onservatoire **N**ational des **A**rts et **M**étiers)
Ce mémoire d'ingénieur décrit l'analyse, la conception et la réalisation d'une interface cartographique d'analyse territoriale multiscalaire de phénomènes sociaux dans le cadre du projet européen Hypercarte pour l'étude et l'aménagement du territoire européen. Il est réalisé au sein du laboratoire **S**ystèmes d'**I**nformation : in**G**énierie et **M**ultimédi**A** du laboratoire **L**ogiciels, **S**ystèmes et **R**éseaux de l'institut d'**I**nformatique et **M**athématiques **A**ppliquées de **G**renoble. Le travail effectué sous la tutelle de M. Jérôme Gensel, Maître de Conférences à l'Université Pierre Mendès France de Grenoble, s'est déroulé du 02 septembre 2002 au 31 août 2003 dans le cadre d'un congé formation.

Remerciements

Je remercie Madame Véronique DONZEAU-GOUGE, Professeur au Conservatoire National des Arts et Métiers de Paris, de présider le jury de ce mémoire.

Je remercie Monsieur Eric GRESSIER, Professeur au Conservatoire National des Arts et Métiers de Paris, d'être membre du jury de ce mémoire.

Je remercie Monsieur Jacques COURTIN, Professeur à l'Université Pierre Mendès France de Grenoble, d'être membre du jury de ce mémoire.

Je remercie Monsieur Jean-Pierre GIRAUDIN, Professeur à l'Université Pierre Mendès France de Grenoble, pour ses conseils lors de l'élaboration de ce travail ainsi que de m'avoir accueilli au sein de son équipe de recherche pour réaliser ce mémoire.

Je remercie Monsieur Hervé MARTIN, Professeur à l'Université Joseph Fourier de Grenoble, Monsieur Jérôme GENSEL, Maître de Conférences à l'Université Pierre Mendès France de Grenoble et Madame Paule-Annick DAVOINE[1], Maître de Conférences à l'Institut National Polytechnique de Grenoble pour leur encadrement et leur aide précieuse tout au long de l'année.

Je tiens à remercier également :

Toute l'équipe SIGMA[2] du laboratoire LSR[3] de l'IMAG[4] ainsi que son équipe administrative pour cette année passée ensemble, et en particulier Marlène, Bogdan et Saïd.

L'ensemble du personnel administratif du CUEFA pour leur disponibilité et leur efficacité.

[1] Mme Paule-Annick DAVOINE possède une double compétence en informatique et en géographie (MC).
[2] SIGMA : Systèmes d'Information : inGénierie et MultimédiA.
[3] LSR : Logiciels, Systèmes et Réseaux.
[4] IMAG : institut d'Informatique et Mathématiques Appliquées de Grenoble.

Table des matières

Table des illustrations

I LE CONTEXTE DU PROJET

I.1 Le projet Hypercarte

Les progrès des technologies liées à l'informatique permettent d'accéder facilement à de nombreuses informations. Les systèmes d'information, généralement utilisés pour mieux les exploiter, tentent de répondre aujourd'hui à des problématiques diversifiées. Les demandes qui sont adressées à la cartographie par les sciences sociales et plus généralement par la société en font partie. Les techniques de la communication apportent une dimension supplémentaire en proposant ces réponses à partir de machines distantes et notamment à travers le Web.

Dans un monde en perpétuel mouvement, afin de faciliter la prise de décision des instances politiques en matière d'aménagement du territoire, il devient nécessaire de se doter d'outils de cartographie interactifs performants pour l'analyse et la représentation des phénomènes économiques et sociaux (densité de population, richesse par habitant, etc.). Les rapports européens [ESPON[5]] soulignent aussi régulièrement l'enjeu politique fondamental d'un accès plus facile à l'information spatiale tant pour les chercheurs et les politiques que pour les citoyens dans ce domaine.

Les atlas papier figent chaque phénomène en une représentation unique. Ils n'offrent pas la possibilité d'introduire en temps réel de nombreuses variantes dans le cadre d'un processus interactif appliqué à la carte qui apporte un grand intérêt en termes d'adéquation aux besoins de l'utilisateur. Les solutions de type Systèmes d'Information Géographique (SIG) qui intègrent une dimension spatiale nécessaire pour la représentation des entités cartographiques, tentent de répondre à cette problématique. Ils sont cependant généralement onéreux et s'adressent plus particulièrement à une certaine élite, capable de les maîtriser et donc d'en tirer partie.

Même si les cartes permettant de rendre compte de la distribution d'un phénomène sont en nombre infini, la réalité géographique peut être appréhendée suivant deux modes d'analyse très différents, réalisés dans un contexte, soit de territorialité, soit de continuité (annexe V.1). Dans un contexte de territorialité, la cartographie qui en résulte utilisera les limites d'un ou plusieurs maillages territoriaux pour visualiser l'information sous la forme de phénomènes discrets (e.g. frontières) (figure V.1.a en annexe). Ces mêmes phénomènes, analysés dans un contexte de continuité, seront restitués dans un espace affranchi de mailles territoriales (figure V.1.b en annexe) à l'aide de procédure de lissage. L'objectif initial du projet Hypercarte [HYPERC] s'inscrit dans cette dernière démarche.
L'étude des différenciations sociales dans le cadre d'un maillage territoriale permettra de représenter un même phénomène (la densité de population, la richesse par habitant, etc.) de façon très différente en fonction des hypothèses sur l'articulation des territoires administratifs (écarts d'une région au niveau européen, au niveau national, au niveau des régions voisines etc.). Tandis qu'une étude des différenciations sociales fondée sur un lissage spatial des distributions permettra de repérer de nouvelles formes sociales correspondant au phénomène observé (e.g. quantité de population accessible en fonction de la distance) grâce à de nouveaux outils et concepts d'analyse spatiale.

C'est dans ce contexte que le projet Européen de recherche Hypercarte est né en réseau avec des chercheurs de l'ensemble des pays de l'Union européenne, ainsi qu'avec des liens prévus avec les instituts statistiques nationaux (INSEE), européens (EUROSTAT, EEA) et mondiaux (PNUD,

[5] ESPON : European Spatial Planning Observation Network.

Banque Mondiale). L'objectif général du projet est d'une part, de mettre au point des outils interactifs de production, de représentation et d'interrogation cartographique des phénomènes sociaux qui prennent en compte l'infinie variété et l'extrême complexité des demandes sociales et politiques qui peuvent être adressées à la carte. D'autre part, il vise à faciliter un accès des citoyens à l'information statistique à travers le Web.

Comme il existe beaucoup de variantes possibles permettant de rendre compte de la distribution d'un phénomène, il est nécessaire de définir des familles de cartes en fonction de leur mode de représentation spatiale (maillage ou surface continue), de la nature du phénomène observé (stock ou taux), du mode d'intégration dans le temps (statique ou dynamique) et enfin du type de démarche souhaitée (descriptive ou prospective). Ces différentes familles de cartes devront être analysées et réalisées au sein du projet Hypercarte à partir de plusieurs modules présentés ci-dessous :

- Le Module d'Analyse Territoriale Multiscalaire (ATM) [ATM03] visera à établir des cartes observant la distribution de l'activité sociale dans les limites d'un ou plusieurs maillages territoriaux que l'on suppose pertinents par rapport aux phénomènes ou aux interrogations qui lui sont adressées par la société. Ce module constitue l'objet de ce mémoire.

- Le Module d'Analyse Spatiale Multiscalaire (ASM) portera au contraire sur l'analyse des distributions de phénomènes sociaux à l'aide de fonctions de lissages fondées sur des voisinages spatiaux multiscalaires.

- Le Module d'Analyse Dynamique et de Prospective (ADP) visera à exploiter des bases de données diachroniques, éventuellement hétérogènes, en vue de la reconstitution de dynamiques spatio-temporelles et de l'établissement de scénarios prospectifs.

A ce jour, de nombreuses réalisations du module ASM ont été produites [HYPERC] afin de valider la dimension opérationnelle et la mise au point de nouveaux concepts et de nouveaux outils d'analyse spatiale des formes sociales.

Conjointement à cette appréhension continue (accessibilité) d'un phénomène, il est nécessaire de répondre favorablement aux attentes plus classiques d'appréhension discrète d'un phénomène (territorialité) pour les utilisateurs dans le domaine de l'aménagement du territoire. C'est un des objectifs du module ATM qui vise à produire des indicateurs couramment utilisés et reconnus (e.g. écart des régions à la moyenne européenne, à la moyenne nationale, etc.). La création de ce module permettra ainsi de diffuser plus largement la plateforme Hypercarte.

Des applications thématiques de ces modules seront fournies sur CD-ROM ou serveur Web, afin de répondre à des demandes sociales ou politiques précises, notamment dans le domaine de l'aménagement du territoire européen ou dans celui de l'étude des inégalités économiques et sociales au niveau mondial.

C'est dans le cadre du projet Hypercarte afin de réaliser le module ATM que ce stage est effectué sous la responsabilité de Jérôme Gensel, Paule-Annick Davoine et Hervé Martin en collaboration avec d'autres équipes partenaires du projet.

I.2 Les équipes et la répartition des tâches

Le module ATM doit permettre la production d'un environnement cartographique capable de répondre, en termes d'analyse et de représentation, aux phénomènes socio-économiques que l'on cherche à étudier. Pour y parvenir, l'environnement devra proposer un ensemble de cartes interactives et adaptées à l'utilisateur par le biais d'une interface Web reliée à un serveur. La conception de ce module s'inscrit donc clairement dans le cadre d'une démarche pluridisciplinaire entre la géographie, les sciences sociales et les sciences et techniques de l'information et de la communication.

Ainsi, la définition des phénomènes potentiellement intéressants, la mobilisation des outils d'analyse et la proposition des interprétations des phénomènes sociaux à représenter relèvent plus directement de la responsabilité de l'UMR[6] Géographie-cités (*Equipe Paris*).

La mise au point d'une infrastructure de calcul et de stockage apte à répondre à un flux de demandes élevé pour de grands volumes de données tout en assurant un temps de réponse rapide relève davantage de la responsabilité de l'équipe informatique et de calcul parallèle ID[7]-IMAG (*Equipe Evaluation de performance et débogage*).

Les problèmes informatiques posés par l'accès à une information virtuellement infinie relèvent quant à eux de l'expertise de l'équipe de système d'information (LSR-IMAG, *Equipe Sigma*) 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'ambiguïté ou d'erreurs d'interprétation de la part de l'utilisateur.

Suivant ces grands axes de compétences, différentes tâches sont spécifiées pour concourir au développement du module (figure 1).

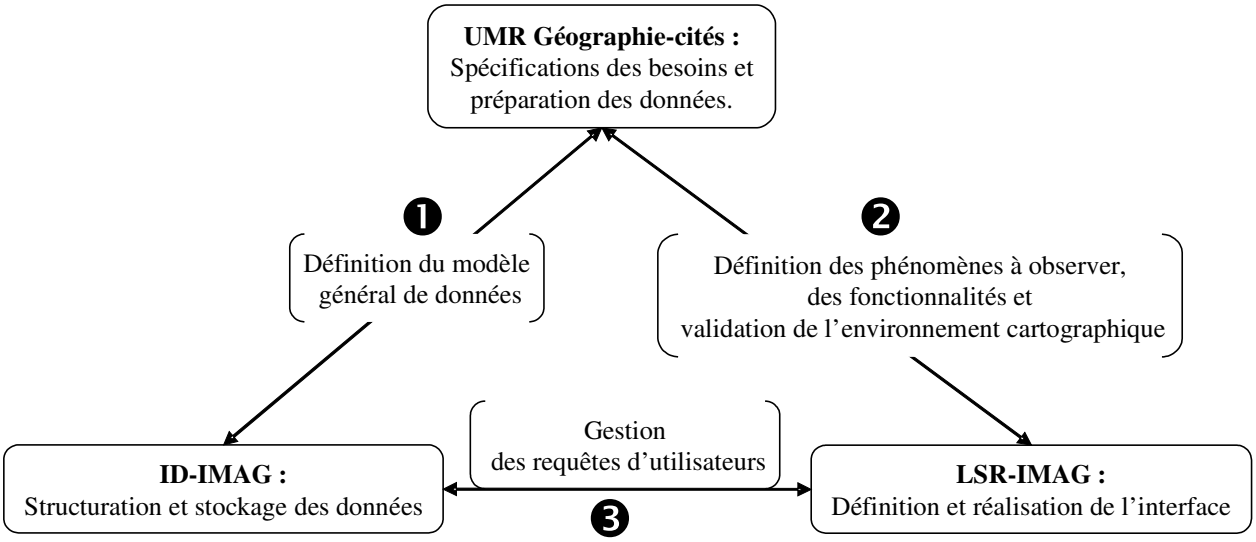


Figure 1 : Des équipes et des tâches.

- L'équipe du laboratoire Géographie-cités spécifie les besoins en termes de résultats escomptés. Pour se faire elle fournit une maquette de l'interface Web côté utilisateur et des spécifications cartographiques détaillées de l'environnement cartographique à

[6] UMR : Unité Mixte de Recherche (Géographie-cités : UMR8504).
[7] ID : Informatique et Distribution, (UMR5132).

développer. L'équipe met aussi à disposition les données socio-économiques, sous la forme de fichiers au format texte, en vue de leur exploitation pour la mise au point du prototype. Enfin, des cartes (figées) représentant les phénomènes à observer seront remises pour évaluer et comparer les résultats obtenus avec notre solution informatique.

- L'équipe du laboratoire ID-IMAG assure la mise à jour et le stockage des données sur le serveur en optimisant les délais de réponses des requêtes client. La structuration finale des données retenues devra faire partie du modèle général de données ❶ établit conjointement avec l'équipe du laboratoire Géographie-cités. Cette définition du modèle permettra d'assurer la « compatibilité » avec les nouvelles données qui seront fournies par les géographes et leur stockage sur le serveur.

- L'équipe du laboratoire LSR-IMAG doit réaliser une interface cartographique interactive et ergonomique accessible pour l'utilisateur par le biais d'un navigateur Web. L'environnement à produire tiendra compte des spécifications des géographes pour la représentation cartographique des phénomènes à observer (analyse) et des fonctionnalités associées à l'environnement (interactivité) ❷. L'environnement cartographique sera validé pendant tout le processus de développement par les géographes ainsi que par notre équipe. L'interface utilisateur côté client sera à terme couplée avec le serveur pour utiliser son jeu de donnée via des requêtes ❸.

La réalisation de cette interface que l'on pourra qualifier d'interface cartographique d'analyse territoriale multiscalaire fait partie d'un des thèmes de recherche et développement de l'équipe SIGMA du laboratoire LSR de l'IMAG. C'est au sein de cette équipe, en collaboration très étroite avec les autres protagonistes cités que ce stage est effectué.

I.3 Module d'Analyse Territoriale Multiscalaire et interface utilisateur

L'objectif du Module d'Analyse Territoriale Multiscalaire consiste à établir des cartes en observant la distribution spatiale de phénomènes sociaux selon un ou plusieurs maillages territoriaux que l'on suppose pertinents par rapport aux phénomènes que l'on cherche à étudier.

Ce module d'analyse doit intégrer une approche de type multiscalaire pour répondre plus spécifiquement à des fins politiques comme quoi il ne faut pas évaluer dans l'absolu mais en termes relatifs [GRASL00]. En effet, les analyses hors contexte, souvent présentées dans les documents européens, mènent à une vision territoriale partielle de l'information [ATM03]. La connaissance d'un indicateur socio-économique comme le taux de chômage par habitant pour une région donnée, apporte certes une information mais isolée de son contexte. Le politique qui détient des responsabilités au niveau régional sera intéressé par cet indicateur mais en le comparant avec les régions voisines à la sienne. L'interprétation de cette mesure pouvant présager d'un flux potentiel légitime de population.
Les analyses multiscalaires tentent de remédier à ces attentes. Elles peuvent porter sur l'analyse d'indicateurs à des niveaux de découpage de l'espace différent. Ces niveaux de découpage sont décrits à partir de la Nomenclature des Unités Territoriales Statistiques (NUTS). Définie pour tous les pays, la NUTS précise pour chaque niveau, le découpage (maillage) qui lui correspond. Pour la France, le niveau 0 correspond au pays, le niveau 1 correspond aux ZEAT[8], le niveau 2 correspond aux régions etc.). Une analyse de type hiérarchique pourrait ainsi comparer un indicateur statistique entre une région et son secteur national de référence (le pays). Dans un

[8] ZEAT : Zones d'Etude et d'Aménagement du Territoire.

même ordre d'idée, il devrait être possible de calculer les écarts relatifs entre un pays et un espace plus important comme l'Europe des 15 (non défini dans la NUTS).

D'autres analyses peuvent compléter la connaissance d'un phénomène dans son contexte. La comparaison, toujours pour un indicateur donné, entre une région et ses voisins repose sur des critères de proximité. Si les voisins de la région concernée partagent une frontière avec celle-ci, nous parlerons plus précisément de voisinage de type contiguïté ou de déviation locale.

Toutes ces analyses (ou déviations) permettent de rendre compte avec précision d'un phénomène étudié dans son contexte. La production de ces informations « efficaces » d'un point de vue politique doit désormais être visualisée et appréhendée par l'utilisateur sans ambiguïté. Un défi important du programme doit remédier à cette situation en proposant des typologies efficaces avec tous les niveaux de déviations considérées (Europe, nation, région, localité). Ces analyses pourront être analysées séparément ou en association pour représenter la synthèse des déviations.

Pour répondre aux attentes en matière d'analyse et de représentation cartographique de l'ensemble des phénomènes sociaux à observer, l'environnement cartographique devra conduire à la définition de différentes familles de cartes. Il devra aussi intégrer la demande d'interactivité en offrant à l'utilisateur différentes fonctionnalités de traitement cartographique.
Ces attentes ont été synthétisées et proposées par l'équipe du laboratoire Géographie-cités comme maquette de base de l'interface pour l'environnement cartographique du côté client (figure 2).

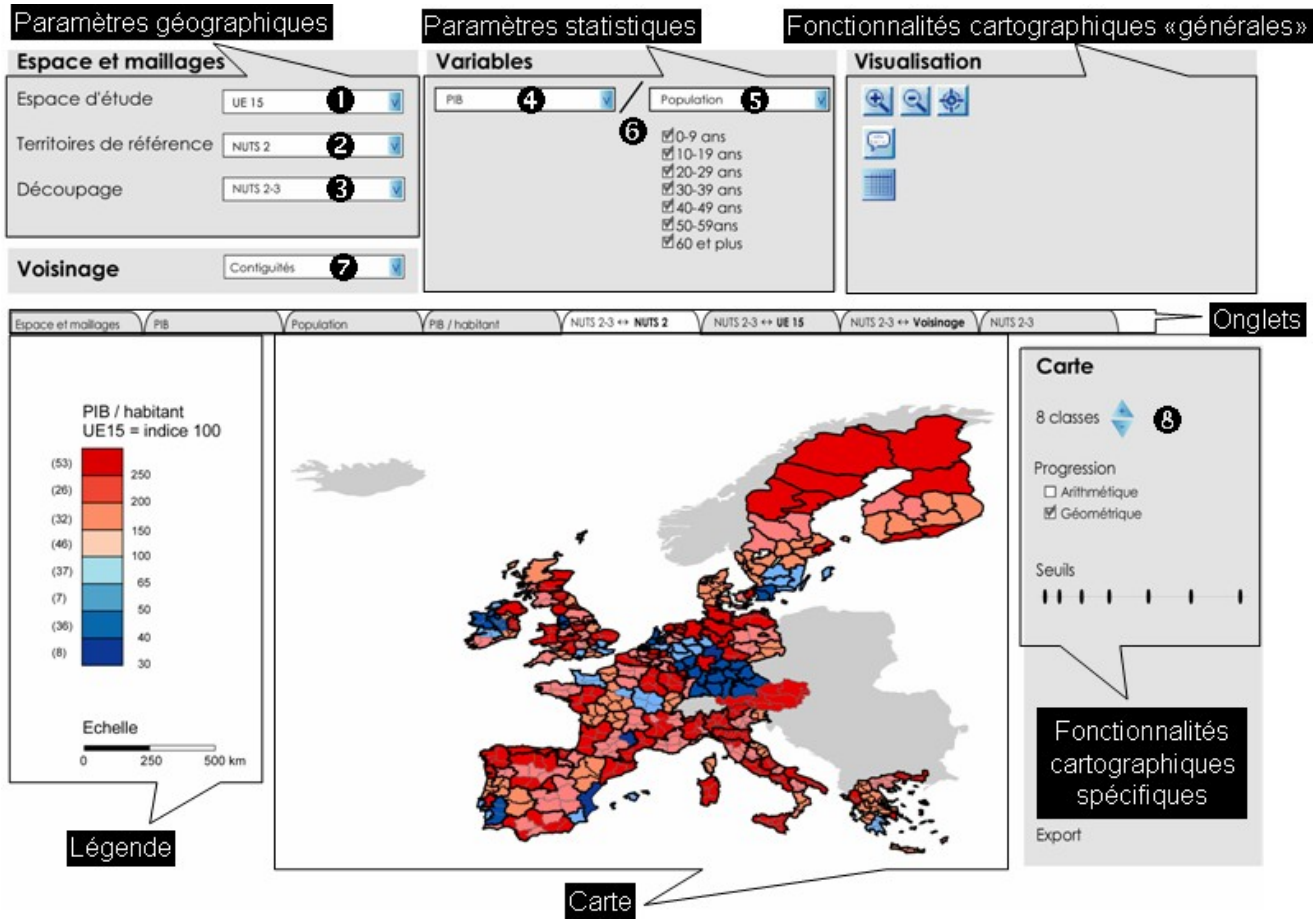


Figure 2 : Première maquette de l'interface utilisateur du module ATM.

Afin de visualiser une carte, l'environnement cartographique devra permettre à l'utilisateur de choisir les paramètres géographiques de sa requête. Il indiquera, par exemple, comme espace

d'étude « UE 15 » ❶ pour l'Europe des 15 correspondant à l'espace qui sera analysé et représenté en zone non grisée. Il renseignera ensuite le découpage (géographique) ❸ qui correspond au maillage souhaité pour être représenté. Sur notre maquette ce découpage correspond au NUTS 2-3 (e.g. les départements). Les frontières de ce maillage seront affichées avec un trait fin. Les territoires de référence ❷ (e.g. NUTS 2 pour les régions) seront quant à eux représentés avec un contour plus épais pour les différencier.

L'espace de travail et les différents maillages établis, l'utilisateur sélectionnera ensuite le ou les paramètres statistiques à prendre en compte (par exemple, les variables PIB[9] ❹ et population[10] ❺) et le calcul éventuel à leur appliquer (par exemple, le rapport «/» ❻). L'analyse portant sur l'indicateur « PIB/Population » pourra ainsi être effectuée.

Les différents paramètres géographiques et statistiques sélectionnés par l'utilisateur, pourront être envoyés au serveur qui retournera ensuite les données correspondant aux indicateurs demandés.

Ces données permettront la représentation cartographique des différents phénomènes à observer. A chaque représentation correspondront une carte, sa légende et des fonctionnalités cartographiques spécifiques à celle-ci.
Les différentes cartographies des phénomènes seront accessibles à l'utilisateur par le biais d'onglets disponibles sur l'interface utilisateur. Cette liste d'onglets est présentée de gauche à droite (figure 2 «Onglets») :

- l'onglet « espace et maillages » représentera une carte correspondant aux seuls critères géographiques sélectionnés (espace d'étude, territoire de référence et découpage) à l'aide de frontières aux contours plus ou moins épais.
- les onglets « PIB » et « Population » représenteront deux cartes à l'aide de cercles proportionnels, par leur surface, à la valeur de l'indicateur représenté, respectivement le PIB et la population.
- l'onglet « PIB/habitant » représentera le ratio des deux variables (PIB et population) sous la forme de quantile[11].
- l'onglet « NUTS 2-3 ←→ NUTS 2 » représentera la déviation (analyse) de l'indicateur considéré (le ratio PIB/population) entre les niveaux NUTS 2-3 et NUTS 2.
- l'onglet « NUTS 2-3 ←→ UE 15 » analysera le phénomène mais entre les niveaux NUTS 2-3 et UE 15.
- l'onglet « NUTS 2-3 ←→ Voisinage » représentera l'analyse des unités de niveau NUTS 2-3 pour l'indicateur PIB/Population par rapport à ses régions voisines contiguës (voisinage de type contiguïté ❼),
- l'onglet « NUTS 2-3 » présentera la cartographie de plusieurs phénomènes combinés. Ainsi, le ratio concerné (PIB/Population) de chaque unité territoriale de niveaux NUTS 2-3 sera analysé en fonction de son territoire de référence, de l'espace d'étude et enfin de son type de voisinage (ici la contiguïté). Cette représentation constituera donc une synthèse des analyses.

Pour toutes les cartes produites, l'environnement cartographique doit offrir des options à l'utilisateur. Ces options peuvent être spécifiques pour chaque phénomène étudié (fonctionnalités

[9] PIB : Produit Intérieur Brut.
[10] La population pourra être analysée plus finement grâce aux tranches d'âges.
[11] Quantile : classes de même effectif.

cartographiques spécifiques). Elles permettront de procéder à des traitements cartographiques divers. Ainsi, la sélection du nombre de classes ❸ affichera, par exemple, la carte en huit couleurs. Ces couleurs figureront aussi sur la légende associée à la carte.

De façon similaire, l'environnement cartographique attendu devra être doté de fonctionnalités traditionnellement offertes pour ce type de produit. Elles seront disponibles pour l'ensemble des représentations. Ces interactions avec l'utilisateur désignées dans notre maquette par fonctionnalités cartographiques «générales» permettront, par exemple, de zoomer sur la carte, de se déplacer dans celle-ci, ou encore d'accéder à des informations complémentaires comme le nom de l'entité sélectionnée. Mais d'autres interactions seront à étudier. Ainsi, le survol sur une entité cartographique avec une souris pourra afficher le nom et les indicateurs de celle-ci sous la forme d'une « info bulle ». Tandis qu'un « clic » sur une entité cartographique affichera sous forme d'histogramme ses différentes analyses.

Notre travail, dans le cadre de ce module, consistera donc à analyser et représenter les différents phénomènes retenus car jugés potentiellement intéressants. Ces analyses pourront être établies séparément ou en association (e.g. synthèse des déviations). Les phénomènes seront observés par l'intermédiaire d'une interface cartographique Web reliée avec un serveur (e.g. échanges). L'interface attendue devra être ergonomique et posséder de nombreuses fonctionnalités d'interaction avec l'utilisateur. L'interface utilisateur côté client pourra être validée par l'intermédiaire des cartes à produire.

I.4 Cahier des charges et problèmes soulevés

Afin d'obtenir des résultats en adéquation avec le but recherché (les phénomènes à analyser et représenter), il est nécessaire de faire un état des lieux de l'existant. Ainsi, les fichiers de données mis à notre disposition seront succinctement décrits. Cette description sera précédée par les diverses notions et précisions utiles à la compréhension sous la rubrique vocabulaire. Les représentations envisagées pour les cartes et les diverses interactions attendues par le système seront précisées à partir de plusieurs documents de travail portant sur les spécifications cartographiques et les fonctionnalités. Cet état des lieux conduira naturellement à la problématique abordée.

Vocabulaire

Les entités traitées au niveau informatique sont des Unités Territoriales (UT). Elles s'organisent en niveaux, formant un découpage (une partition) de l'espace. Chaque UT sera référencée dans la NUTS (utilisée pour les statistiques de l'Union Européenne) quand il existe un niveau correspondant.

Exemple complet pour la France :

NUTS **0** Pays
NUTS **1** ZEAT (Zones d'Etudes et d'Aménagement du Territoire)
NUTS **2** régions
NUTS **3** départements
NUTS **4** *inexistant* (mais correspond, par exemple, aux districts en Grande Bretagne)
NUTS **5** communes

Les différents découpages s'organisent selon une structure hiérarchique emboîtée. On doit pouvoir passer d'un niveau de découpage plus élevé à un autre moins élevé (exemple : le passage du département à la région).

Les UT qui composent le niveau le plus bas de la hiérarchie sont appelées Unités Territoriales Elémentaires (UTE).

Fichiers de données disponibles

Différentes informations sont fournies par l'équipe du laboratoire Géographie-cités sous la forme de fichiers de données en format texte. La structure de ces fichiers sera vue comme une contrainte à respecter. Ces données seront une base de notre travail pour réaliser le module ATM.

L'Unité Territoriale (UT) compose le lien central entre les différents types d'information que nous allons manipuler. Elle permettra de définir différents types de relations.

La base d'informations territoriales nous est fournie dans le fichier *UT.txt* qui contient l'ensemble des unités territoriales, tous niveaux confondus. Il est décrit au moyen de deux champs, le *Code_UT* et le *Nom_UT* correspondant respectivement à l'identifiant de l'unité territoriale et à son nom.

La base d'informations territoriales est caractérisée par des relations hiérarchiques et des relations d'appartenance.

- Le fichier *UT_SUP.txt* décrit la hiérarchie d'emboîtement des UT au moyen de deux champs, l'identifiant de l'UT et l'identifiant de son UT supérieure (*Code_UT* et *Code_UT_Sup*).

- Le fichier de description de l'appartenance de chaque unité à un espace et à un découpage s'appelle *Espace.txt*. Il est composé de dix champs :
  Code_UT, UE15, Europe_Elargie, Arc_Atlantique, PECO, NUTS4, NUTS3, NUTS2, NUTS1, et NUTS0.
  La notion d'appartenance est codée par un « 1 ».

Exemple : Le Code_UT qui correspond à l'entité France « appartient » à l'Europe des 15 (e.g. UE15) ainsi qu'à l'Europe élargie (e.g. Europe des 25). Il fait partie du découpage en NUTS0 car c'est un pays. Les colonnes UE15, Europe_Elargie et NUTS0 correspondant à l'UT France seront donc renseigné par un « 1 ».

La base d'informations territoriales comporte (e.g. «est en relation avec») des données statistiques.

- Pour notre étude, les fichiers *Pib99.txt* et *Population99.txt* sont composés de deux champs, le *Code_UTE* et l'*Attribut* qui correspondent respectivement à l'identifiant de l'unité territoriale élémentaire et à la valeur statistique à considérer (le PIB en 1999 et le nombre d'habitants en 1999). Il y aura donc autant de fichiers de données statistiques que nécessaire.

La base d'informations territoriales comporte aussi (e.g. «est en relation avec») des données de géométrie qui permettent une représentation dans l'espace (visualisation).

- Chaque unité territoriale est associée à un ou une série de polygones, représentés par une suite de points. Ainsi, le fichier *map.txt* décrit pour chaque UT, le ou les contours des polygones ainsi que le numéro de polygone correspondant. Les informations sont données sous la forme *Code_UT*, *X*, *Y*, et *Num_Polygone*, avec *X* et *Y* les coordonnées de latitude et longitude non projetées de chaque point composant le polygone, et *Num_Polygone* correspond au numéro de polygone de l'UT.

- L'information géographique est également donnée par le fichier *Centroide.txt* qui définit la géométrie des centroïdes correspondant aux UT. Les informations de ce fichier sont de la forme *Code_UT, X, Y*, avec X et Y les coordonnées géométriques données en latitude et longitude non projetées correspondant au centroïde de l'UT.

Remarque : il n'existe qu'un centroïde par UT, même si celle-ci est composée de plusieurs polygones.

Spécifications cartographiques et fonctionnalités

Le cahier des charges met à disposition un ensemble de spécifications cartographiques et de fonctionnalités détaillées. Ces spécifications rendent compte de la sémiologie cartographique à respecter ainsi que les diverses fonctionnalités attendues par le système.

Un extrait (couvrant un large éventail) des spécifications qui concernent l'ensemble des cartes du module est fourni ci-dessous :

- Une carte représente une variable (par exemple, le PIB) par l'intermédiaire de cercles.
- Une carte représente le ratio de deux variables à l'aide d'une gamme de couleurs avec une seule dominante (par exemple, un dégradé de bleu).
- Les limites (frontières) des unités correspondant au niveau découpage sont représentées à l'aide d'un trait fin noir (0.12 point)
- On accède aux cartes par le biais d'onglets disposés sur l'interface.
- Si le bouton « information » des outils de visualisation est sélectionné, lorsque l'utilisateur fait passer sa souris sur la carte, une « info bulle » apparaît et indique le nom de l'unité et la valeur de la variable ou de l'indice représenté.

Une synthèse des différentes spécifications cartographiques est donnée en annexe V.2. Cette synthèse, proposée par famille de carte, montre plus en détail l'ensemble des contraintes liées à l'application pour l'interface utilisateur du côté client.

Problématique

Si les problèmes soulevés par l'environnement cartographique à développer sont nombreux, nous pouvons cependant mettre en évidence quelques points remarquables. Ces difficultés concernent plus particulièrement la structuration et l'accès à l'information, la représentation cartographique ainsi que les échanges entre machines distantes.

Pour analyser des phénomènes d'emboîtement hiérarchique, on doit se demander, par exemple, comment une UT peut accéder à son UT de niveau englobant supérieur. La problématique visée sera donc comment structurer et représenter cette hiérarchie.

L'analyse multiscalaire fait aussi intervenir des phénomènes de voisinage territorial de type contiguïté. Contrairement aux phénomènes d'emboîtement hiérarchique, cette analyse ne sera effectuée que sur un seul niveau de découpage. Il faudra identifier, pour chaque unité territoriale, les UT qui lui sont contiguës d'un point de vue géométrique. Aucune information dans les fichiers de données n'apportant ce type d'information, il sera donc nécessaire de se demander comment identifier et représenter cette notion de contiguïté.

Le recueil des données statistiques n'est fourni qu'au niveau de l'unité territoriale élémentaire. Pourtant, les analyses hiérarchiques doivent comparer les données socio-économiques à deux niveaux de découpage. On se demandera donc comment affecter ces données à partir du niveau le plus bas de la hiérarchie (lieu de collecte des informations statistiques) sur l'ensemble des unités composant cette hiérarchie.
Dans un même ordre d'idée, les analyses peuvent faire intervenir de nouveaux indicateurs socio-économiques comme le ratio PIB/habitant, créé à partir des données statistiques PIB et population auxquelles on affectera l'opérateur « / » pour conduire au ratio souhaité. Dans ce cas, l'affectation des données statistiques au sein de la hiérarchie ne relèvera plus d'une simple addition de variables simples. Il sera donc nécessaire de s'interroger comment effectuer des analyses sur des données dérivées (e.g. ratio de deux variables).

L'environnement cartographique souhaité doit permettre de représenter sur une même carte des UT appartenant à des niveaux de NUTS différents. Les régions pourront, par exemple, être caractérisées avec une frontière au contour épais et les départements avec un contour plus fin. Il faudra donc s'interroger sur le langage ou la technologie adaptée capable de restituer la distinction entre ces différents niveaux de maillage.

Les problématiques liées à la structuration des données et leurs relations éventuelles, ainsi que celles relevant de l'affectation des données statistiques manquantes sont posées. Il reste à définir les problèmes soulevés par la restitution de l'information cartographique.

Tout d'abord, pour l'ensemble des cartes considérées, en termes de représentation de la donnée géographique, nous devons nous demander sous quelles formes restituer la carte à l'utilisateur (image de type bitmap ou vectoriel) et quelle technologie utiliser.

D'autres problèmes plus spécifiques à la conception et à la réalisation de l'interface utilisateur côté client peuvent être mis en évidence. Il faut s'assurer que la maquette proposée permet bien de restituer l'ensemble des phénomènes à considérer et procéder au choix de l'environnement informatique à utiliser pour le restituer.

Il faudra veiller à ce que le résultat cartographique attendu ne soit pas source d'erreurs d'interprétation par l'utilisateur. La carte étant, par essence, graphique, elle constitue un mode particulier de diffusion de contenus qui renvoie à la problématique générale de la présentation. On peut se demander quels sont les critères les plus pertinents à retenir pour représenter un phénomène et les proposer éventuellement comme valeurs par défaut (nombre de classes de couleurs, type de progression arithmétique etc.). Doit-on associer à la donnée cartographique produite des informations complémentaires comme l'aide en ligne, etc. Certains éléments de réponses pourront être appréhendés grâce aux spécifications cartographiques qui constitueront un réseau de contraintes à respecter.

A l'environnement cartographique, il faut associer la carte qui peut être considérée comme un composant interactif grâce à des fonctionnalités qui lui sont propres (zoomer, déplacer, etc.) posant la problématique des transformations géométriques à visualiser dans l'espace.
La carte possède aussi un contenu. Ainsi, une entité territoriale sélectionnée avec la souris fera apparaître diverses informations la concernant. Mais comment peut on identifier l'unité à considérer quand plusieurs niveaux de découpage sont affichés ?

Une autre forme de problématique concerne les échanges entre machines distantes. Quelles sont les informations disponibles pour le client au lancement de l'application ? Sous quel format transmettre ces informations ? Et enfin il convient d'aborder la problématique du « qui fait quoi ?» afin notamment d'optimiser les procédures de calcul et d'échange de données entre machines distantes.

En conclusion, si la conception d'une application cartographique «classique» reste une tâche complexe, le prototype d'analyse territoriale multiscalaire intègre des problématiques générales supplémentaires. Ainsi, l'accessibilité à partir du Web, la possibilité d'analyser des phénomènes contextuels et enfin la représentation d'un environnement cartographique interactif seront des difficultés à surmonter.

I.5 L'organisation du mémoire

Ce mémoire s'organise en 3 chapitres.

Le premier chapitre a présenté, en guise d'introduction, le projet Hypercarte dans sa problématique générale. La réalisation des sous objectifs a été partitionnée en 3 modules dont celui de l'analyse territoriale multiscalaire qui constitue l'objet de notre étude. Celui-ci est décomposé en tâches afférentes aux différentes équipes partenaires du projet pour conduire à l'objectif final qui sera présenté : l'interface utilisateur côté client. Cette réalisation repose sur un cahier des charges (les fonctionnalités attendues, l'ergonomie) et un existant (les fichiers sources) qui induisent à la problématique spécifique déjà décrite.

Le deuxième chapitre introduit les concepts liés aux systèmes d'information géographiques permettant de situer et d'appréhender certaines spécificités relatives au domaine concerné. Un éventail de solutions susceptibles de satisfaire nos exigences est proposé. Il concerne plus particulièrement trois sous objectifs de notre problématique, à savoir la réalisation d'analyses spatiales, les architectures capables de répondre favorablement à la diffusion cartographique interactive sur Internet. Et enfin les langages graphiques permettant d'afficher la donnée cartographique sur notre interface utilisateur.

Le troisième chapitre explicite et synthétise les différents choix de conception adoptés pour notre réalisation. Notre projet de développement est ensuite modélisé grâce à un formalisme orienté objet. Puis intervient la réalisation proprement dite, observée à partir de productions cartographiques (e.g. des copies d'écran) accompagnées de commentaires.

Enfin, un bilan du travail est présenté en guise de conclusion à travers une évaluation des résultats obtenus au regard des contraintes et des objectifs initiaux soulevés au chapitre I.4 (section : problématique soulevée). Un ensemble de perspectives venant s'inscrire dans la continuité de ce travail est également proposé.

II ETAT DE L'ART

La première partie de ce chapitre présente, après un bref historique et diverses définitions, les notions relatives aux Systèmes d'Information Géographiques. Les principales fonctions d'un SIG, les modes de représentation de la donnée cartographique et les relations sont ainsi abordés. Une attention plus particulière est apportée aux outils d'analyses et notamment d'analyse spatiale pour les SIG vectoriels en deux dimensions, pour amorcer des solutions potentielles à l'un de nos thèmes d'étude. Quelques produits du marché sont ensuite brièvement analysés afin d'en soulever les éventuelles insuffisances pour notre objectif. Une conclusion sous la forme de réponses à nos besoins clôture ce paragraphe II.1.
Cette partie est complétée par la description de solutions susceptibles d'être mises en œuvre dans notre projet. Les architectures et technologies associées capables de satisfaire à la cartographie interactive sur Internet sont ainsi étudiées dans le paragraphe II.2. Tout comme deux « langages » dits ouverts, analysés en parallèle, spécifiques pour la représentation de la donnée cartographique (paragraphe II.3).

II.1 SIG

Les Systèmes d'Information Géographique sont apparus à la fin des années 60 parallèlement aux outils CAD destinés à la production automatisée de cartes et d'outils de bases de données. En 1969, l'Environnement System Research Institute (ESRI[12]), à but non lucratif, est créé et met au point une nouvelle méthode de représentation cartographique basée sur une vision cellulaire des données (i.e. le mode image[13]). Dès les années 80, les premiers outils tel que ArcInfo développé par l'ESRI apparaissent et apportent les premiers formats standardisés de SIG (image et vectoriel).

Le besoin de publier de l'information géographique en ligne et sur le Web est né de la relative maturité des technologies de l'internet et des SIG. La capitalisation d'expérience dans ces deux domaines a permis de satisfaire le besoin de publier des cartes en ligne. Le développement de fonctionnalités interactives à partir d'interfaces graphiques conviviales entre l'utilisateur et les données mises à disposition par le diffuseur ont apporté de la valeur ajoutée à ce merveilleux outil d'analyse.
Car la carte sous sa forme numérique représente un support de communication synthétique et attractif. La représentation cartographique permet de faire interagir des acteurs sur un territoire donné. La spatialisation de la donnée change considérablement la façon d'organiser et d'analyser la gestion d'un territoire, et ceci à n'importe quelle échelle. De plus, la pression citoyenne, les articles et les textes législatifs en faveur de la diffusion de données semblent favoriser une démocratisation de ces outils.

Les domaines d'applications des SIG sont variés et recouvrent des domaines tels que l'aménagement du territoire, l'urbanisme, la gestion des espaces, la gestion des réseaux et des transports, la géomarketing etc.

De nombreux acronymes sont présents dans la littérature. Les plus représentatifs sont les Systèmes d'Information à Références Spatiales (SIRS) ; les Systèmes d'Information sur le Territoire (SIT) qui possèdent généralement une composante spatiale ; les Systèmes

[12] ESRI : http://www.esri.com/
[13] Mode image : image sous la forme de pixels.

d'Information Urbains à Référence Spatiale (SIURS) et les systèmes d'Information Localisée (SIL). Pour des besoins de commodité, nous utiliserons le sigle SIG dans la suite de ce document.

II.1.1 Définitions

Un Système d'Information Géographique ou GIS pour Geographical Information System est défini de façon très générale comme un :

« *système informatisé qui comprend une base de données sur un ensemble d'unités géographiques, et un logiciel ou un ensemble de logiciels permettant le stockage, la mise à jour, un accès efficace aux informations, le traitement et la représentation visuelle de ces données* » [BEGUI94].

L'économiste Michel Didier (1990), dans une étude réalisée à la demande du Comité National de l'Informatique Géographique (CNIG), donne une définition de l'information géographique qui en précise les objectifs. Il s'agit pour l'auteur de mettre à disposition un :

«*Ensemble de données repérées dans l'espace, structuré de façon à pouvoir en extraire commodément des synthèses utiles à la décision ».*

Transposé dans le cadre des SIG, cette définition, peut définir la finalité du système du point de vue de l'utilisateur, comme une modélisation de la réalité utile à la prise de décision. Le monde réel peut être représenté sous la forme de données qui seront numérisées. Cette information numérique pourra être structurée et stockée dans une base de données (BD) afin d'être analysée pour en recueillir de l'information (figure 3).

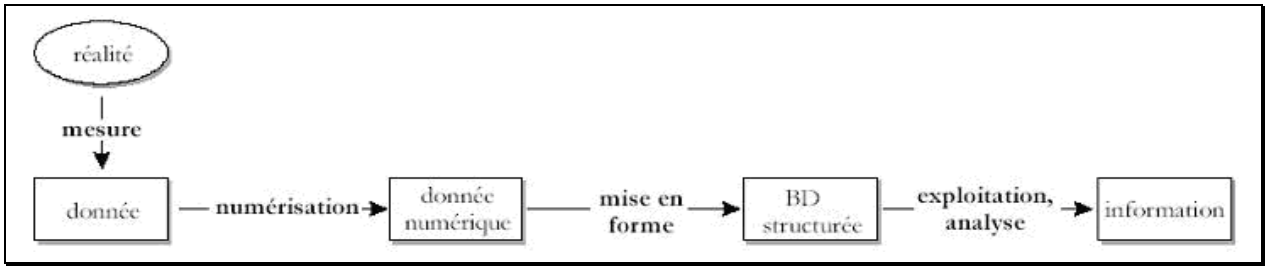


Figure 3 : De la réalité à l'information. [UNIGE]

Il est intéressant de noter, dans la littérature, plusieurs définitions des SIG ayant des philosophies différentes. Par exemple, la définition américaine, émanant du comité fédéral de coordination inter-agences pour la cartographie numérique (FICCDC, 1998) insiste sur les différentes fonctions techniques que doit comporter un SIG :

« *System of computer hardware, software, and procedures designed to support the capture, management, manipulation, analysis, modeling, and display of spatially referenced data solving complex planning and management problems* »

Il est ainsi possible de dégager deux grandes caractéristiques spécifiques des SIG [DENEG96] :

- La capacité de gérer et de traiter les relations spatiales entre objets ou phénomènes dans l'espace terrestre, impliquant des fonctions d'analyse spatiale (non courantes dans les traitements d'information classiques) et de synthèse pour l'aide à la décision.
- La représentation de cet espace, sous la forme d'une carte ou d'un plan, ce qui implique des fonctions de conception et de production cartographiques, constituant en elles-mêmes un langage différent du langage ordinaire (textuel ou numérique).

II.1.2 Principales fonctions d'un SIG : les « 5A »

Les SIG sont des outils informatiques qui permettent de manipuler des données géo-référencées. Les 5 fonctionnalités de base d'un SIG d'après Denègre et Salgé [DENEG96] sont représentées dans la figure 4. On parle alors des « 5A » :

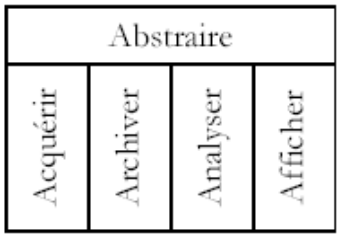


Figure 4 : Schéma de principe des SIG.

- L'acquisition des données (par saisie, digitalisation, importation de données numériques, GPS etc.) consiste à transformer en un format utilisable par le système les différents éléments qui entrent en jeu pour étudier le phénomène choisi.

- L'archivage des données concerne le stockage de ces données généralement sous la forme de bases de données relationnelles. (Il permet le stockage, mais aussi l'interrogation, la mise à jour, l'ajout et la suppression de ces données, le plus souvent à travers un SGBD[14]).

- L'analyse et l'interrogation (requête, modélisation, simulation) permettent notamment d'interroger le système en termes de requêtes spatiales ou attributaires, ainsi que de manipuler le système (projection, changement d'échelles).

- L'affichage des données est rendu possible sur écran ou imprimante. Les données sémantiques sont affichées selon une symbolisation définie par l'utilisateur pour produire une carte qui représente la distribution spatiale de ces données.

- L'abstraction de la réalité (e.g. modèle) doit être prise en considération par les outils SIG.

Remarque : la notion d'abstraction ne représente pas une « fonctionnalité » proprement dite mais plutôt un choix de conception ou de représentation de la réalité.

II.1.3 Représentation et modélisation des données

L'information contenue dans une base de données géographique se divise en deux grandes entités : les informations (ou données) spatiales qui permettent de décrire visuellement sous forme de cartes un phénomène, et les informations de type attributaires ou thématiques qui définissent le champ ou le thème d'application.

II.1.3.1 Données spatiales

La représentation graphique des données spatiales peut se matérialiser suivant deux grands modèles de données : le modèle matriciel (ou raster) et le modèle vectoriel.

II.1.3.1.1 Le modèle matriciel

[14] SGBD : Système de Gestion de Bases de Données.

Le monde réel peut être représenté à l'aide d'une photographie ou d'une carte. La numérisation de cette image produit une matrice de points (i.e. les pixels). La modélisation (structuration) matricielle s'effectue en apposant une grille (régulière) sur la surface à représenter et en attribuant à chaque cellule la valeur dominante ou moyenne de l'attribut se rapportant à cette surface. On parle alors de représentation de type matriciel ou raster (figure 5).

Dans cet exemple, l'attribut considéré peut être un champ, un lac, une rivière ou une habitation. La valeur de l'attribut sera exprimée en fonction de la couleur et de la concentration de pixels associées à chaque cellule. Ainsi, la valeur de l'attribut « 2 » qui correspond aux rivières est attribuée à chaque cellule qui contient quelques pixels de couleur bleue. Les lacs sont codés par des « 1 » car le nombre de pixels de couleur bleue par cellule est plus important. L'habitation est représentée par des pixels de couleur rouge, la cellule correspondante est codé par un « 3 ». Les pixels qui ne correspondent à aucune entité auront quant à eux la valeur « 0 » dans les cellules correspondantes de la matrice résultante. Cette valeur peut être considérée comme un champ de verdure.

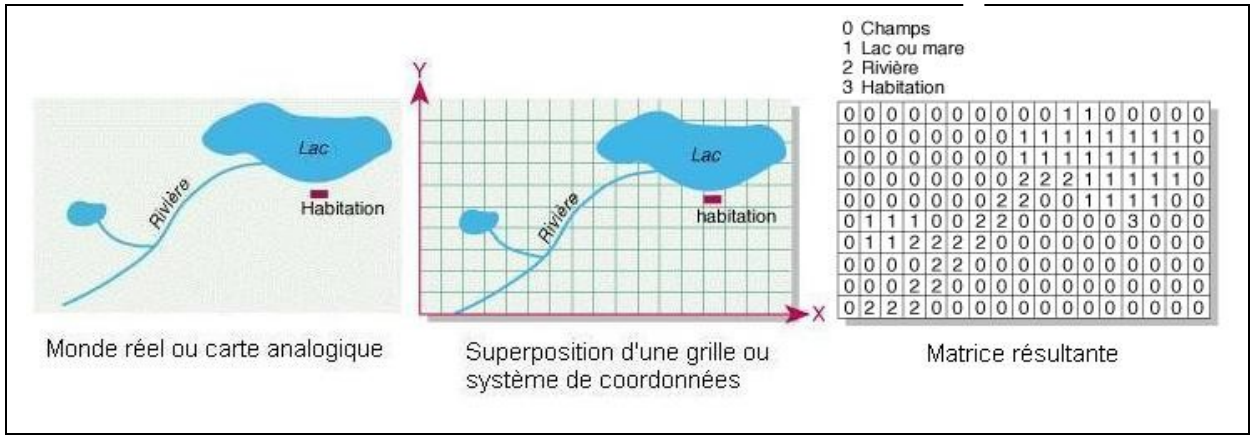


Figure 5 : Le mode matriciel ou raster.

Une donnée raster se compose donc d'une suite de pixels de taille régulière. Chaque pixel porte généralement une information de radiométrie (e.g. la couleur) qui conditionnera la valeur numérique présente dans la matrice résultante. La plupart du temps, ces valeurs reflètent la classe du pixel (par exemple, le type de sol) ou l'envergure d'un phénomène (par exemple, la pente).

II.1.3.1.2 Le modèle vectoriel

Les données vectorielles se présentent sous la forme d'éléments géométriques repérés dans l'espace grâce à leurs coordonnées en x, y et éventuellement z pour la hauteur.
La figure 6 représente le monde réel (identique à l'exemple précédent) sous une forme vectorielle.

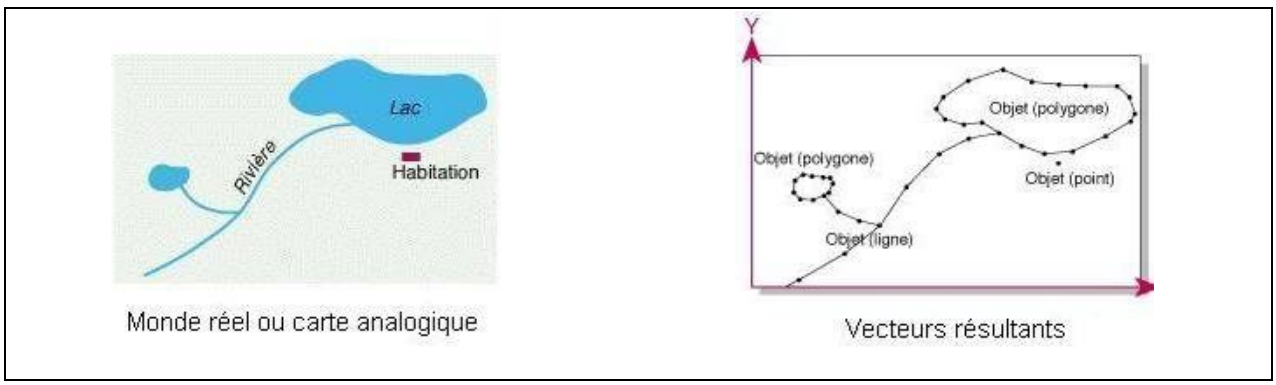


Figure 6 : Le mode vectoriel.

Afin de représenter et décrire les objets du système, il est nécessaire de spécifier un ensemble de primitives et d'objets cartographiques simples à 0, 1 et 2 dimensions pour décrire respectivement des points, des lignes et des surfaces. On distingue ainsi (figure 7) :

- Les objets ponctuels auxquels on associe un seul jeu de coordonnées donnant la position du point dans l'espace, i.e. le point utilisé pour identifier la localisation d'un élément ponctuel comme des tours, des puits, une borne d'incendie, une habitation, etc.

- Les objets linéaires associés à une suite ordonnée de points contigus, chaque point de cette suite est relié au point suivant par un segment de ligne (e.g. le vecteur[15] qui peut représenter une route linéaire). Le phénomène peut se répéter dans une orientation différente de l'espace. Ainsi, l'objet linéaire sera traduit par une succession de lignes brisées représentant le phénomène linéaire (figure 6 : les rivières).

- Les objets surfaciques délimités par un objet linéaire qui se ferme sur lui-même. On peut parler de polygone délimitant l'objet surfacique qui peut représenter une commune. Dans le cas de surface à trous (polygone complexe), le polygone intérieur vient se soustraire à la surface considérée.

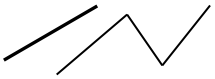
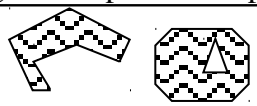
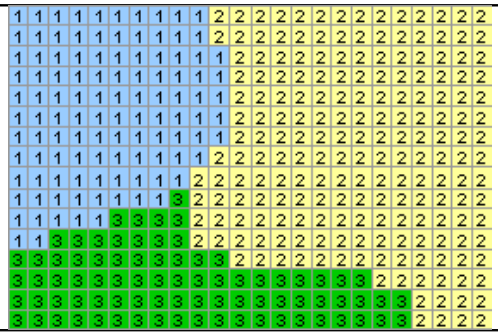
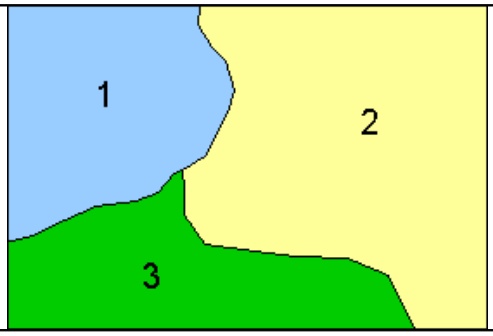
	Objets ponctuels	Objets linéaires	Objets surfaciques
Dimension	Objets à 0 dimension	Objets à 1 dimension	Objets à 2 dimensions
Primitives	Point	Ligne - Polylignes	Polygone simple ou complexe
Exemple	• une borne	 une route	 une commune

Figure 7 : Primitives géométriques.

II.1.3.1.3 Comparaison des technologies vecteur et raster

Les deux modes de représentation offrent chacun des avantages et des inconvénients. Les particularités retenues sont synthétisées dans la figure 8 :

Format ou modèle de données[16]	
Mode matriciel (ou raster)	**Mode vecteur**
	
Une cellule représente toujours une surface	Les coordonnées en x et en y définissent indifféremment des points, des vecteurs et des surfaces.
La collecte de données est rapide	La collecte de données est lente

[15] Vecteur pour le mode vectoriel.

[16] Les images proviennent du site http://www.fao.org

Format ou modèle de données	
Le stockage est volumineux	Le stockage est compact
La structuration des données est simple (les attributs sont rattachés aux cellules)	La structuration des données est complexe (les attributs sont rattachés aux entités)
La résolution dépend de la taille de la cellule	La résolution dépend du nombre de décimales
La qualité graphique dépend de l'échelle de représentation	La qualité graphique est indépendante de l'échelle de représentation
Le croisement thématique est simple	Le croisement thématique est plus complexe
Ce mode est adapté à des données continues	Ce mode est adapté à des objets discrets dont les limites sont précises
Les opérations sur les surfaces sont faciles (calcul d'aire, superposition de couches).	Les opérations portant sur les points et les surfaces sont faciles (système de projection, calcul de périmètre)

Figure 8 : Comparaison des modes raster et vecteur.

Selon la nature de l'information spatiale à traiter, l'un des modes peut sembler plus pertinent que l'autre. Le mode raster offre de nombreux avantages pour la présentation de données continues (couverture végétale, précipitations, densité) alors que le mode vectoriel est plus adapté à des données discrètes dont les limites sont précises (limites administratives, infrastructures, réseaux). Ces deux modes de représentation peuvent être complémentaires. On peut ainsi retrouver des images matricielles superposées à des données vectorielles pour accroître le niveau de détail du fond d'image par exemple.

II.1.3.2 Données thématiques

La donnée attributaire représente souvent une information statistique selon un thème, d'où son appellation de thématique. Les données thématiques ne présentent pas de spécificités particulières. En effet, ces données répondent aux mêmes exigences que les systèmes d'information classiques.

Remarque : Il est nécessaire de préciser qu'un troisième type de données existe : l'information temporelle. Elle fait l'objet, au sein même du laboratoire LSR, de nombreux efforts en termes de recherche et développement. On peut notamment citer le projet européen SPHERE (Systematic Paleoflood and Historical data for the improvEment of flood Risk Estimation) [SPHER99].

II.1.4 Les relations dans un SIG vectoriel

Afin d'effectuer la modélisation complète d'un espace géographique selon les critères retenus, il est nécessaire de décrire explicitement les diverses relations que l'on peut rencontrer dans un SIG [IGNEN] : les relations spatiales, les relations sémantiques, les relations de composition et les relations de construction.

II.1.4.1 Les relations spatiales

Les relations spatiales définissent les liens entre les différentes entités géométriques à représenter. Elles doivent pouvoir décrire si deux routes se croisent (connectivité) ou bien si deux bâtiments sont voisins (adjacence), etc. via les objets géométriques correspondants. Ces entités géométriques, dans la théorie des graphes, se présentent sous l'appellation de faces (e.g. polygones ou surface), d'arcs (e.g. lignes) et de nœuds (e.g. points) (figure 9).

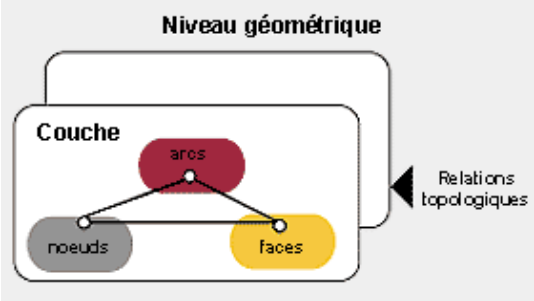


Figure 9 : Information géographique : niveau géométrique.

Deux faces peuvent, par exemple, être contiguës et donc partager au moins un arc. Un noeud peut se trouver à l'intersection de deux arcs qui se croisent etc.
Ces relations spatiales sont aussi appelées relations topologiques. Nous les étudierons plus en détail dans la section II.1.5.2.2.

II.1.4.2 Les relations sémantiques et les relations de composition

Les relations sémantiques décrivent les relations logiques entre objets. L'appartenance d'un bâtiment à un propriétaire, une route franchit telle voie ferrée, etc.
Les relations de composition décrivent comment un objet complexe est composé de plusieurs objets élémentaires.

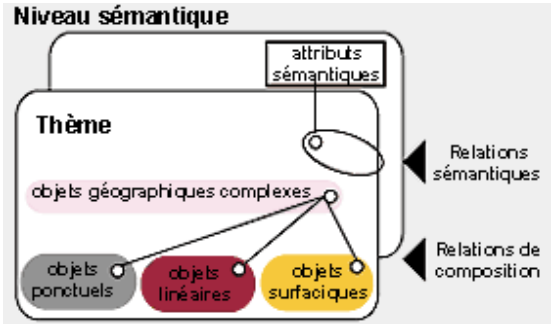


Figure 10 : Information géographique : niveau sémantique.

En fonction de l'échelle de représentation souhaitée, un objet pourra être observé comme un « objet complexe » dans la mesure où il est constitué d'objets plus simples. L'agrégation de plusieurs primitives géométriques pour représenter une entité géographique complexe constitue un objet complexe. Par exemple, une ville est une surface complexe. Elle est composée de nombreux immeubles et routes. Cette ville est un objet élémentaire (e.g. vu comme un point) à l'échelle 1/40 000 000 mais devient un objet complexe à des échelles plus petites[17].

II.1.4.3 Les relations de construction

La correspondance entre l'objet sémantique et sa « traduction » géométrique se réalise grâce aux relations de construction. La modélisation demande que l'on associe un objet graphique, aussi qualifié de cartographique, à l'entité que l'on souhaite représenter. Une habitation peut ainsi devenir un point, une rivière peut devenir une succession de lignes, et un lac peut être représenté par un polygone. Ces relations décrivent les liens entre le niveau géométrique et sémantique par une opération de jointure relationnelle. On parle alors de couplage entre information spatiale et information attributaire que l'on retrouve nécessairement dans les SIG. L'originalité d'un SIG, par rapport à d'autres systèmes, est liée à cette jointure permanente entre informations spatiales et attributaires, qui sont analysées et modifiées conjointement.

[17]Une échelle plus petite correspond à un agrandissement.

L'ensemble de ces quatre types de relation (spatiales, sémantiques, de composition et de construction) permet de modéliser explicitement la base de données comme le montre la figure 11.

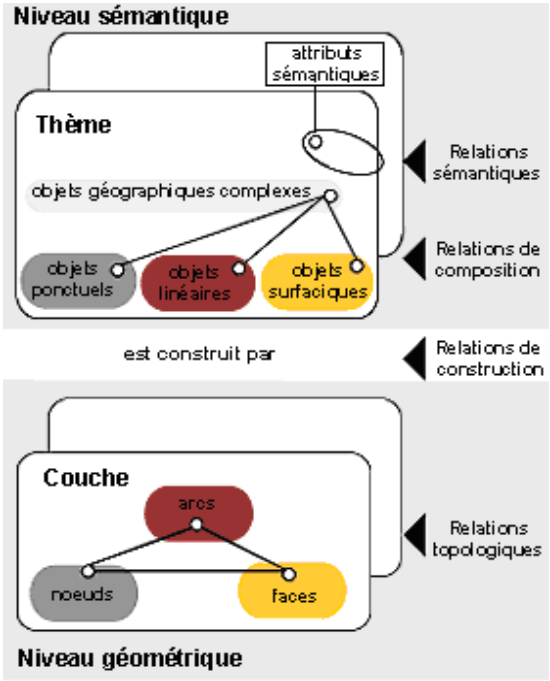


Figure 11 : Le modèle conceptuel de l'information géographique.

Les SIG permettent ainsi « d'intégrer » les données du monde réel, généralement selon des thèmes choisis (démographies, activités économiques, réseaux de transports, etc.), pour visualiser la distribution spatiale de phénomènes sous forme de cartes thématiques. Ces dernières peuvent être superposées, ce qui permet de mettre en relation plusieurs variables et aussi de mieux présenter les scénarios et solutions qui facilitent la compréhension et la prise de décision. On définit ainsi une couche comme un plan réunissant des éléments géographiques de même type. Une couche peut aussi être vue comme un compartiment logique du système d'information. Ainsi chaque couche représente un sous-ensemble « thématique » des informations retrouvées dans le SIG.

II.1.5 Les outils d'analyse

Les SIG offrent bien sûr de nombreuses fonctions d'analyses et donc de traitement de l'information. Albrecht [ALBRE95] en recense environ 150 qui pourront être classifiées selon le type de données traitées (vecteur ou raster), ou le domaine d'appartenance des données (thématiques et/ou spatiales), ou encore le niveau d'analyse (modèle à topologie ou non). Ces fonctions sont basées sur des requêtes portant sur les attributs des objets, leurs relations sémantiques, topologiques et de composition.

Les outils d'analyse les plus fréquents établissent des requêtes sur des bases de données, le plus souvent spécialisées pour la géographie.
Une base de données est une "*structure de données permettant de recevoir, de stocker et de fournir à la demande des données à de multiples utilisateurs indépendants*" (définition AFNOR-ISO, dictionnaire de l'informatique, 1989).
Ces bases de données seront interrogées généralement avec le langage de requêtes SQL (Structured Query Language), basé sur l'algèbre relationnelle, afin d'effectuer les opérations souhaitées.

De nombreuses requêtes peuvent s'appliquer à un SIG mais deux grandes familles se dégagent : les requêtes thématiques et les requêtes spatiales.

II.1.5.1 Requêtes thématiques

Les requêtes thématiques peuvent porter sur des critères descriptifs variés d'ordre quantitatif ou qualitatif. Par exemple, qu'elle est la densité de population pour la zone géographique sélectionnée ? Ou, qu'elle est la répartition de la population en classes sociales pour cette zone ?

Les requêtes thématiques permettent de sélectionner un sous-ensemble de la base de données à partir de critères attributaires. Ces traitements peuvent aussi être effectués dans des bases de données non géographiques (traitements classiques dans les systèmes d'information).

Exemple de traitement thématique pour connaître le nombre d'habitants (e.g. population) du département de l'Isère à partir d'une base de données *Departement* ayant comme attributs son *nom* et sa *population* :

> *SELECT Departement.population*
> *FROM Departement*
> *WHERE Departement.nom = "Isere"*

Une base de données peut aussi être analysée graphiquement par la mise en place de légendes visibles sur l'interface utilisateur. Les légendes thématiques obtenues ont un lien dynamique avec la table permettent, par exemple, d'établir une légende pour chaque attribut de la table. L'avantage majeur tient du fait qu'une mise à jour de la table entraîne une mise à jour automatique de la carte (et donc de sa légende).

II.1.5.2 Requêtes et opérations spatiales

Il n'est pas aisé de décrire l'ensemble des opérations qui concernent les objets spatiaux. Cependant, une classification logique peut être établie [SCHOL02] :

- Les opérations ensemblistes (union, intersection...)
- Les transformations géométriques (translation, rotation...)
- Les opérations de calcul (longueurs, aires, périmètre, centroïde...)
- Les opérations topologiques (inclusion, adjacence...)

De nombreuses opérations faisant intervenir l'analyse spatiale sont directement liées à la modélisation rencontrée au niveau des SIG. C'est pourquoi il est important de décrire tout d'abord les principaux modèles rencontrés. L'analyse qui en résulte (relations spatiales entre les objets géométriques) conditionne généralement les opérations qui peuvent lui être adressées.

II.1.5.2.1 Modèle sans relation spatiale entre les objets géométriques

En l'absence de structuration, il est tout à fait possible d'afficher, par exemple, les contours d'une carte en deux dimensions. En effet, un simple fichier en format ASCII ne comprenant qu'une succession de coordonnées en x et en y définira la position des points dans l'espace. Ces points seront ensuite reliés pour former des segments de droite et enfin des contours (e.g. surfaces). Les formes obtenues seront ainsi affichées.

Si nous pouvons observer cette carte qui relève plus du dessin cartographique, elle ne pourra en aucun cas être analysée spatialement ou sur la base de ses attributs. C'est pourquoi, dans de

nombreux systèmes géo-informatiques en technologie vectorielle, les entités géographiques sont représentées de manière individuelles et indépendantes.

Le format général de ces entités se présente généralement sous la forme d'un identifiant, du nombre de coordonnées (1 pour un point, 2 pour une ligne et au moins 2 pour un polyligne) suivi de l'ensemble des coordonnées en x, en y et éventuellement en z pour l'altitude. Une liste d'attributs pourra également être affectée pour chaque entité géographique (figure 12).

	Entité ponctuelle	**Entités linéaires**	
	Point	Lignes	Polylignes/Polygones
coordonnées dans l'espace des entités géographiques	x y (z)	x (début) y (début) (z début) x (fin) y (fin) (z fin)	Chaîne (x,y,(z))
	attribut 1 attribut 2 (…)	attribut 1 attribut 2 (…)	attribut 1 attribut 2 (…)

Figure 12 : Eléments géométriques avec coordonnées et attributs.

Remarque : le polygone est parfois représenté par une suite de coordonnées (polylignes) dans laquelle la dernière est la répétition de la première, pour "boucler la boucle" (c'est le cas par exemple des logiciels de SIG MapInfo[18] et Idrisi[19]).

Le format « spaghetti » ne prend pas en compte les relations entre les entités géographiques (appartenance, contiguïté, etc.). Ce mode de structuration vectorielle reflète donc une absence de structuration logique, c'est un modèle non topologique (modèle implicite), qui occasionne des SIG dit "en mode objet". Les données vectorielles se présentent telles que numérisées et suivant la position des différents objets dans l'espace, les lignes peuvent se recouper ou se juxtaposer sans cohérence apparente.
Ainsi, s'il est possible d'afficher et de comprendre une carte de type « spaghetti » à l'écran, celle-ci ne pourra généralement pas être interrogée et analysée (par exemple : « quels sont les polygones à la droite de *x* ? »). Cette lacune incite à explorer les modèles qui, explicitement, prennent en compte les relations entre les objets géométriques.

II.1.5.2.2 Modèles explicites : la topologie

La topologie désigne l'expression des relations entre les objets géométriques. Elle peut aussi être décrite comme une structuration des données spatiales ou, plus généralement, définie comme une géométrie sans coordonnée (figure 13)

[18] MapInfo : http://www.mapinfo.com
[19] Idrisi : http://www.clarklabs.org/

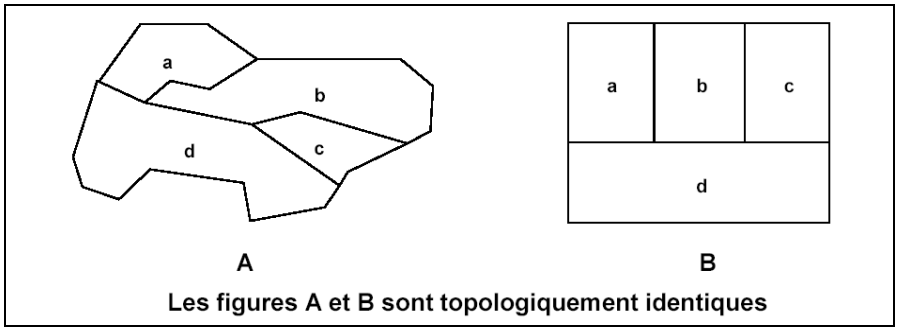


Figure 13 : La topologie.

Les SIG qui gèrent la topologie (comme la gamme ArcGIS d'ESRI), identifient tous les objets les uns par rapports aux autres (le sens de la saisie est, par exemple, enregistré).

La structuration topologique implique, dans la plupart des cas, trois types d'objets :

- les points avec les coordonnées X et Y sans signification topologique mais porteurs de la référence spatiale.
- les noeuds et leurs identifiants, porteurs de la connectivité. On retrouve un nœud à l'intersection des lignes qui se croisent.
- les identifiants des polygones de droite et de gauche pour identifier les voisins.

La topologie doit être étudiée en amont pour que les traitements effectués sur la base de données correspondent à l'expression des besoins de l'utilisateur. La topologie permet aussi d'effectuer des traitements que l'on pourra qualifier d'agrégations, c'est-à-dire visant à fusionner (visuellement) des objets géographiques.

En gardant notre objectif d'affichage de cartes multi niveaux, comme nous possédons l'ensemble des données de géométries des unités territoriales, il sera facile de les exploiter (e.g. les afficher). Cependant il est important d'expliciter la notion d'agrégation des données géographiques qui permettrait avec un seul jeu de données (au niveau élémentaire) et d'une hiérarchie de reconstituer l'ensemble des données de géométrie des UT, sans duplication de frontière (et donc d'information).

La figure 14 montre un exemple simple d'agrégation géométrique. Si le niveau N correspond à un affichage des départements français, le « passage » à une représentation régionale (niveau N-1) devrait se matérialiser comme la figure de droite.

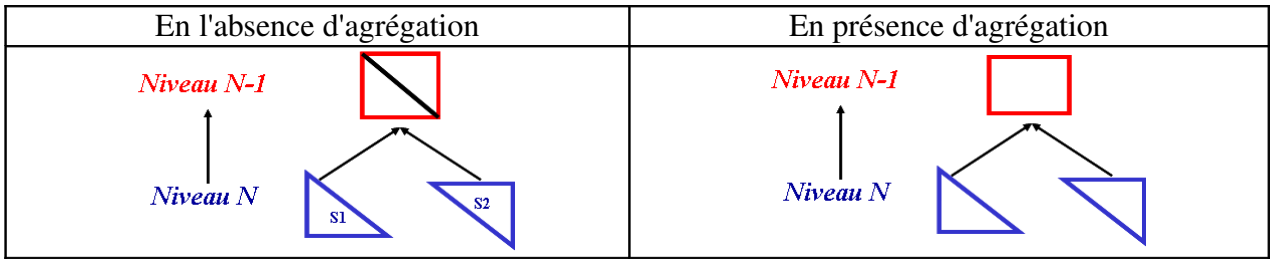


Figure 14 : Agrégation des données géographiques.

Remarque : Les deux surfaces au niveau N sont contiguës.

En absence d'agrégation géographique, le problème peut se poser de la manière suivante :

- La représentation en niveau N-1 ne peut distinguer celle de niveau N

- L'affichage du segment (en noir sur la figure) se fait N fois, deux fois dans notre cas (par *S1* et *S2*).
- Ce même segment ne peut être traité de manière autonome.

Il faut donc introduire une structuration logique capable (dans cette configuration) d'afficher les segments (et donc les points) une seule fois (pas de redondance).
On a ainsi recours à différents modèles topologiques qui s'appuient sur des structures de données [ONGE95] adaptées à la spécificité de la situation. Ils seront « interrogés » tout d'abord grâce à SQL afin d'expliciter le pourquoi de cette modélisation.

Remarque : les modèles à topologie présentés par la suite ont parfois des données attributaires. Elles sont introduites pour identifier le lien possible entre les données attributaires et les données spatiales.

II.1.5.2.2.1 Modèle de topologie nodale

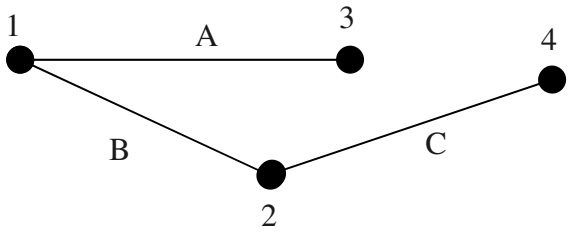
Le modèle de topologie nodale ou modèle réseaux convient particulièrement pour représenter les réseaux linéaires qui peuvent être orientés ou non.

La topologie des réseaux décrit la relation entre des ensembles linéaires grâce à leurs nœuds qui se trouvent à chaque extrémité.
Dans les exemples suivants, les lignes sont identifiées (e.g. *Id_Chaîne*) dans la table dénommée « Chaîne_Segment ». Chaque ligne possède un nœud de départ (e.g. *Nœud_Initial*) et un nœud d'arrivée (e.g. *Nœud_Terminal*) permettant de connaître la relation entre deux arêtes (ou ligne) ainsi qu'éventuellement son sens.

Un réseau non orienté et ne comportant qu'une seule voie comme le réseau de sentiers pédestres pourrait avoir la structure suivante :

> *Chaîne_Segment (Id_Chaîne, Noeud_Initial, Noeud_Terminal)*

La table « Chaîne_Segment » contient l'ensemble des *Chaîne_Segment* à décrire et identifier par le champ « Id_Chaîne ». Les champs « Nœud_Initial » et « Nœud_Terminal » sont les identifiants des points correspondants donnés en coordonnées métriques (non présenté dans la figure 15[20]).



Chaîne_Segment		
Id_Chaîne	Nœud_Initial	Nœud_Terminal
A	1	3
B	1	2
C	2	4

Figure 15 : Topologie d'un réseau non orienté.

On peut circuler à travers le réseau en trouvant les *Chaîne_Segment* dont un des noeuds possède le même identifiant que celui du nœud ou l'on se trouve sur la *Chaîne_Segment* que l'on vient de traverser. Pour trouver les *Chaîne_Segments* rattachées à un noeud dont le numéro serait contenu dans la variable numéro, on pourrait procéder de la façon suivante :

> *SELECT Id_Chaîne*

[20] On recherche ici à montrer les possibilités liées à la topologie de réseau.

FROM Chaîne_Segment
WHERE Noeud_Initial = numéro
OR Noeud_Terminal = numéro ;

L'application récursive d'une requête semblable permet par exemple de « naviguer » à travers un réseau hydrographique.

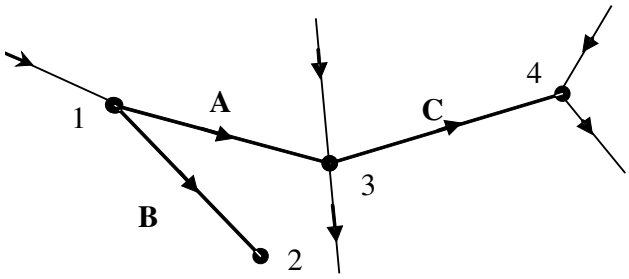
Plusieurs systèmes d'information géographique offrent des possibilités de recherche de parcours optimaux (plus court chemin) ou de circulation à travers le réseau.

Les réseaux orientés sont des réseaux simples dans lesquels la circulation le long d'une *Chaîne_Segment* ne peut se faire que dans un seul sens comme un aqueduc par exemple.
Dans ce cas, le réseau pourrait avoir la structure suivante :

Chaîne_Segment (Id_Chaîne, Noeud_Initial, Noeud_Terminal, Sens_Normal)

Ce sens devra être indiqué implicitement ou explicitement. Plusieurs logiciels retiennent le sens de numérisation (qui se traduit par l'ordre d'enregistrement des coordonnées dans le fichier) et l'utilise comme donnée d'orientation. Il est toutefois préférable de spécifier l'orientation de façon explicite.

La figure 16 donne un exemple de cette orientation donnée explicitement avec :
si *Sens_Normal* est vrai ("TRUE") la direction va du noeud initial vers le noeud terminal,
si *Sens_Normal* est faux ("FALSE") la direction va du noeud terminal vers le noeud initial.



Chaîne_Segment				
Id_Chaîne	Nœud _Initial	Nœud _Terminal	Sens _Normal	Qualité
A	1	3	True	chaussée
B	2	1	False	chaussée étroite
C	3	4	True	chaussée

Noeuds	
Id_Noeud	Qualité
1	croisement
2	Cul-de-sac
3	pont
4	croisement

Figure 16 : Topologie de réseaux orientés avec données attributaires.

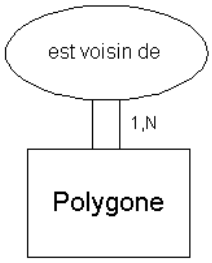
Le champ qui renseigne sur la direction peut être considéré comme une optimisation. En effet, si l'orientation d'un nœud vers un autre doit être changé, seul le champ « Sens_Normal » sera modifié, les identifiants des nœuds ne subiront pas de changement.

En prenant en considération l'orientation (e.g. on ne circule pas dans un réseau en sens inverse de celui prévu), la requête se présenterait ainsi :

SELECT Id_Chaîne
FROM Chaîne_Segment
WHERE (Noeud_Initial = numéro AND Sens_Normal = TRUE)
OR (Noeud_Terminal = numero AND Sens_Normal = FALSE);

II.1.5.2.2.2 Modèle de topologie de surface

La topologie de surface réfère à l'information explicite qui renseigne sur l'identité des voisins d'un polygone. Il existe théoriquement plusieurs façons d'exprimer la relation de voisinage immédiat entre deux polygones. La manière la plus intuitive de représenter cette relation peut se décrire de la manière suivante :



L'entité géométrique surface est représentée par un polygone, identifié par « Id_Polygone ». Cette surface peut contenir des attributs comme la superficie en km^2 etc.
La table « Voisin » permet d'identifier avec le champ « Id_Poly_2 », l'ensemble des polygones qui sont voisins du polygone retenu, identifié par *Id_Poly* comme suit :

> *Polygone (Id_Poly, Surface, etc.)*
> *Voisin (Id_Poly_1, Id_Poly_2)*

```
SELECT Voisin.Id_Poly_2
FROM Polygone, Voisin
WHERE Polygone.Id_Poly = Voisin.Id_Poly_1;
```

Plus généralement (et plus élégamment), la topologie de voisinage permet à partir des arcs (ou arêtes) constituant le polygone de connaître les voisins de chaque surface, comme le montre la figure ci-dessous.

Cette structure peut se traduire de la manière suivante :

> *Chaîne (Id_Chaîne, Poly_Gauche, Poly_Droite)*
> *Polygone (Id_Poly, Surface, etc.)*

Des déclinaisons sont cependant possibles, si l'on cherche à décrire plus précisément les différents attributs pour les objets géographiques concernés et l'orientation du polygone :

> *Chaîne (Id_Chaîne, Longueur, etc.)*
> *Polygone (Id_Polygone, Surface, etc.)*
> *Chaîne_Polygone (Id_Chaîne, Id_Polygone, Côté)*

Avec l'attribut Côté qui peut prendre les valeurs "gauche" ou "droit".

Pour sélectionner les polygones voisins du polygone 14, on écrit alors :

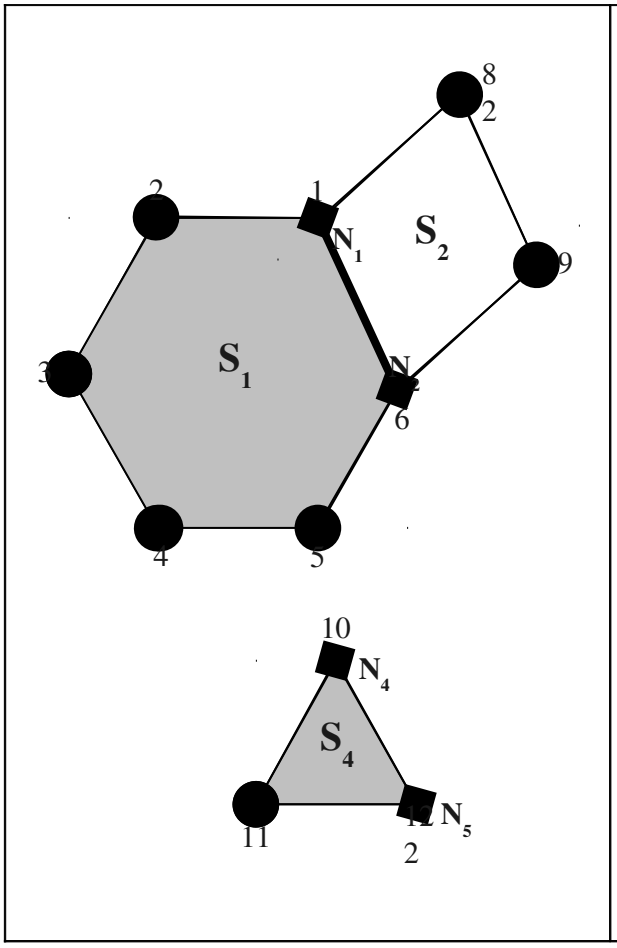
```
SELECT Id_Polygone FROM Chaîne_Polygone
WHERE Id_Chaîne IN (SELECT Id_Chaîne FROM Chaîne_Polygone WHERE
                    Id_Polygone = 14) AND Id_Polygone <> 14;
```

Avec l'opérateur <> qui signifie "différent de".

II.1.5.2.2.3 Modèle hybride

Le modèle hybride présenté est celui utilisé dans *ArcInfo*. Il permet une analyse topologique de réseau et de voisinage. Il sera plus particulièrement développé afin d'étudier les différentes requêtes et opérations spatiales qui peuvent être appliquées au modèle. Il nous servira de support pour répondre à certaines problématiques induites par notre projet.

On distingue les points (porteurs de la référence spatiale) des nœuds (porteurs de la connectivité). Les lignes ou arêtes sont définies une seule fois. Elles sont définies par une séquence de points et délimitées par des nœuds. Les surfaces sont définies par une séquence d'arêtes. Et les zones avec enclaves sont définies par une addition de surface.



Objets

No	Surfaces
1	**1**,4
2	2

Surfaces

No	Séquence arêtes	Couleur
1	**1, 2**	**Gris**
2	2, 3	Blanc
3		
4	5, 6	Gris

Arêtes

No	Sommets	ND	NF	SG	SD	Type
1	1,2,3,4,5,	1	2	1	0	Fin
2	**6, 1**	**2**	**1**	**1**	**2**	**Epais**
3	1, 8, 9, 6	1	2		2	Fin
5	10, 11, 12	4	5	4		Fin
6	12, 10	5	4	4		Fin

Nœuds

No	Pt
1	**1**
2	**6**
4	10
5	12

Points

No	x	y
1		
3		
4		
…		

Figure 17 : Topologie « hybride » de réseaux et de surface.

Explication :
L'objet géographique que l'on souhaite représenter, par exemple une unité territoriale, est mis en relation avec des objets géométriques. Une unité composée de 2 surfaces S_1 et S_2 peut être ainsi associée à l'identifiant *No 1* de la table des *Objets*. La surface S_1 est identifiée dans la table des *Surfaces* par son identifiant *No 1*. Elle comporte la donnée attributaire *Couleur Gris* et est composée de 2 *Séquences d'arêtes* identifiées par *No 1* et *No 2* dans la table des *Arêtes*. L'arête *No 2* est composée des sommets 6 et 1, représentés dans la figure par *N1* et *N2* et reliés par un *Type* de trait *Epais* (attribut Type de la table des arêtes).
Les sommets permettent de matérialiser visuellement (e.g. référence spatiale) les points affichés grâce à la table des *Points* et de leurs identifiants. Les autres attributs de la table des *Arêtes* informe sur la topologie de l'arête : le nœud de départ (e.g. *ND*), le nœud final (e.g. *NF*), la surface à gauche (e.g. *SG*) et la surface à droite (e.g. *SD*).
Les attributs *ND* et *NF* sont en relation avec la table des *Nœuds*, elle même en relation avec la table des *Points*.

Ce modèle est évalué d'une part à travers les requêtes et opérations spatiales évoquées en début de section II.1.5.2 et d'autre part en fonction de nos objectifs de réalisation.

- **Requêtes d'analyse spatiale : les opérations ensemblistes**

L'intersection et l'union réfèrent ici à la théorie des ensembles et aux principes de l'algèbre combinatoire. La relation d'intersection utilise des algorithmes de superposition et de scission de points, de lignes et de polygones pour délimiter les parties du territoire ou identifier les objets qui possèdent les caractéristiques visées par une requête.
Cependant, grâce à la topologie, une opération ensembliste d'union peut s'apparenter à la notion d'agrégation géométrique. Appliqué à notre projet et dans la perspective de posséder un seul jeu de données au plus bas niveau, on recherche à agréger les entités géographiques présentées sous la forme d'un découpage administratif en départements vers un découpage en régions de ces mêmes entités (figure 18).

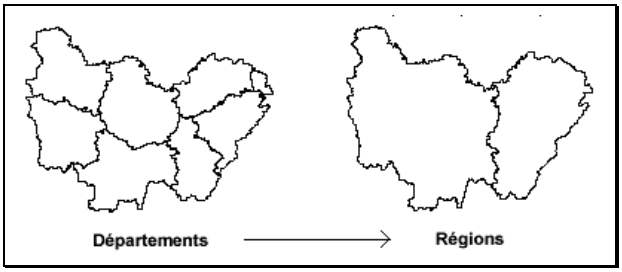


Figure 18 : Agrégation géographique.

Si l'on prend l'hypothèse que les surfaces S_1 et S_2 sont les représentations géométriques de départements qui appartiennent à une seule région, l'agrégation consistera à afficher les surfaces *S1* et *S2* sans l'arête délimitée par les nœuds *N1* et *N2* qu'ils ont en commun.
On considère la table des surfaces et l'on ne sélectionnera que les séquences d'arêtes qui ne sont pas en commun. Ainsi, la surface N° 1 ne contiendra plus que la séquence d'arête « 1 » et la surface N° 2 la séquence d'arête « 3 ». La séquence d'arête « 2 » ne sera donc plus représentée.
A contrario, la séquence d'arête « 2 » peut être représentée avec un trait plus fin afin de la différencier des contours des régions obtenues avec un trait, par exemple, plus épais.

En extrapolant ce type de requête uniquement spatiale vers une requête d'agrégation spatio-thématique, on cherche à opérer des transformations pour les deux types de données en présence, c'est-à-dire les données thématiques et les données géométriques.

L'opération d'agrégation thématique consiste à cumuler les informations, de type attributaire, de niveau *N* vers le niveau supérieur *N-1* comme le montre la figure 19, puis généraliser à tous les niveaux de la hiérarchie[21]. Si la modélisation le permet, une « simple » requête de sélection et de mise à jour permettrait de créer de nouveaux attributs [DAO03] : la somme des départements pour chaque région considérée (figure 20).

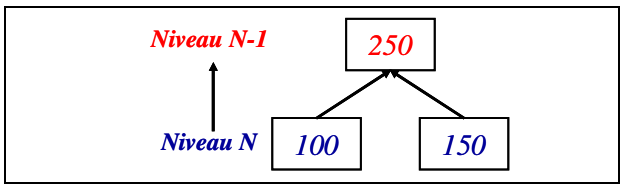


Figure 19 : Agrégation des données thématiques

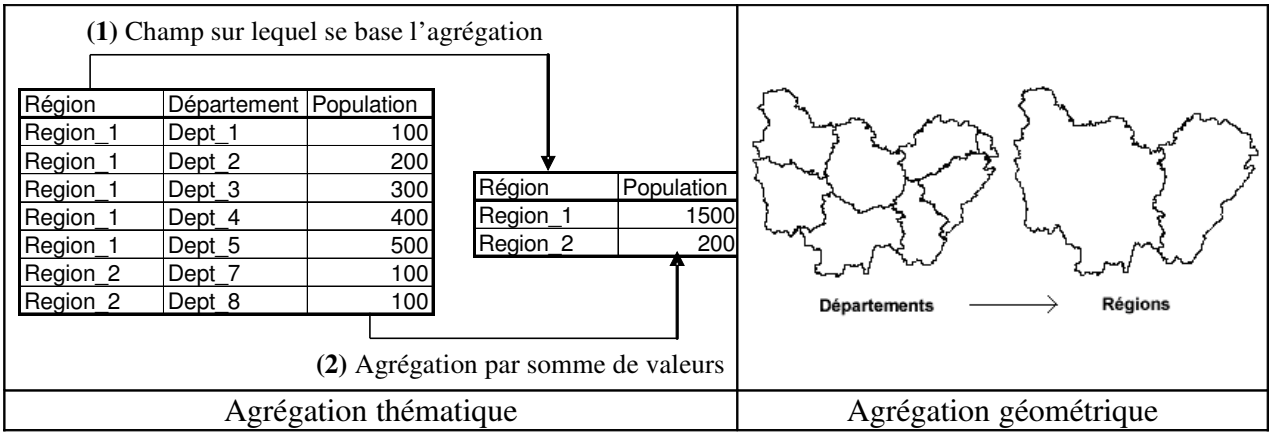


Figure 20 : Agrégation simultanée des données thématiques et géométriques.

Ce type de requêtes exploite simultanément les dimensions spatiales et attributaires des objets.

- **Requêtes d'analyse spatiale : les transformations géométriques**

Une transformation telle qu'une translation permettant de déplacer une carte à l'écran ne fera intervenir que la table des *Points*. Une translation de gauche à droite (axe des X) sera obtenue en affectant la valeur considérée comme nouvelle valeur pour l'attribut *X* dans la table « Points ». Par exemple, X prend la valeur de X auquel on ajoute une variable.

- **Requêtes d'analyse spatiale : les opérations de calcul**

Les opérations qui concernent le calcul d'une surface, d'un périmètre ou encore d'un centroïde, ne sont pas liées à la modélisation du système, mais repose sur une branche des mathématiques, spécialisée dans les algorithmes et les traitements géométriques [MOTET97]. Ces opérations annexes sont généralement proposées aux SIG sous la forme de fonctions complémentaires.
Il est cependant possible, avec le modèle présenté, d'opérer d'autres formes d'opération. Par exemple, si une donnée attributaire indique une longueur dans la table *Arêtes*, il devient possible d'opérer des calculs du plus court chemin, comme le ferait plus spécifiquement une topologie de réseau orienté (section II.1.5.2.2.1).

- **Requêtes d'analyse spatiale : les opérations topologiques**

[21] Dans le cadre de notre projet (problématique).

Les opérations topologiques sont directement influencées par la nature de l'analyse topologique concernée. Nous avons déjà rencontré plusieurs exemples permettant de connaître le voisin d'un polygone, si deux lignes se croisent, etc.

Ce modèle semble bien s'adapter aux différentes analyses que l'on peut lui adresser. Il permet de mettre en évidence la relative complexité, en terme de relations, qu'engendre un simple affichage de surface. Cette « cascade » de relations entraîne nécessairement un impact sur les performances du système.

II.1.5.2.2.4 Comparatif avec et sans topologie

La notion de topologie est essentielle pour des analyses spatiales. Le tableau figure 21 propose un comparatif des différentes structurations rencontrées.

Structure non topologique	Structure topologique
Structure simple Facilité de transferts de données (données « brutes ») Suffisant pour la cartographie numérique Analyse spatiale très réduite Doublement des limites de polygones adjacents	Structure complexe Relations spatiales introduites lors de la saisie Indispensable pour la gestion de réseaux Analyse spatiale performante Aucun dédoublement, ni d'information, ni de dessin, même pour des couches différentes

Figure 21 : Comparatif avec et sans topologie.

II.1.5.2.3 Requêtes avec un modèle relationnel étendu

Le langage SQL s'est imposé comme le langage de requêtes standard des SIG. Il permet d'interroger les différents modèles topologiques observés qui peuvent reposer sur des SGBD non spécialisés. Cependant, si le SGBD relationnel est bien adapté pour les informations de niveau sémantique ou attributaire, les requêtes des SIG utilisent généralement des opérateurs spatiaux portant sur les objets géographiques.

On ajoute ainsi au SGBD relationnel de nouveaux attributs satisfaisant un type abstrait de données (TAD), parmi les plus représentatifs, on peut citer :

- les TAD géométriques (point, ligne, surface)
- les TAD topologiques (arc, noeud)

Le TAD géométrique surface [SCHOL02] spécifié ci-dessous est donné à titre indicatif et permet, pour nos exemples, de s'abstraire du mode de représentation (vecteur ou raster) ainsi que de la structure de la surface (polygone).

Exemple de spécification d'un TAD surface :

> *Type surface*
> *PointInPolygon : surface x point : bool (test l'appartenance d'un point dans une surface)*
> *Contiguë : surface x surface : bool (test la contiguïté entre deux surfaces)*

- **Traitement spatio-thématique**
  Exemple de requête (liste des départements contigus au département de l'Isère) :

> *SELECT d.nom*

FROM Departement d, Departement d'
WHERE d'.nom = "Grenoble" AND Contigue(d.geometrie, d'.geometrie)

- **Traitement spatial interactif**
  Exemple de requête (nom du département pointé à la souris) :

  SELECT Departement.nom
  FROM Departement
  WHERE PointInPolygon (Departement.geometrie, pointClic)

L'implémentation des TAD sera effectuée et disponible au sein de bibliothèques de fonctions. Ces fonctions peuvent être établies grâce à la branche des mathématiques spécialisée dans les algorithmes et traitements géographiques qui couvrent un grand éventail d'opérations. Parmi lesquelles [MOTET97], le calcul d'intersection de 2 segments, le test du point à l'intérieur d'un polygone, la superposition de polygones, etc.

Ces bibliothèques de fonctions sont accessibles via un SQL étendu, comme *SpatialWare* de *MapInfo* [GOLCA01] capable d'utiliser les opérateurs spatiaux du système. On présente, ci-dessous, ceux utilisés par le logiciel *MapInfo* [BARBI02] :

- *Contains* : l'objet A contient l'objet B si le centroïde de B se trouve dans le polygone de A.
- *Contains entire* : l'objet A contient entièrement l'objet B si le polygone de B est entièrement inclus dans le polygone de A.
- *Within* : l'objet A est dans l'objet B si son centroïde est dans le polygone de B.
- *Entirely within* : l'objet A est entièrement dans l'objet B si le polygone de A est entièrement dans le polygone de B.
- *Intersects* : l'objet A rencontre l'objet B si ils ont au moins un point en commun.

II.1.6 Quelques produits du marché

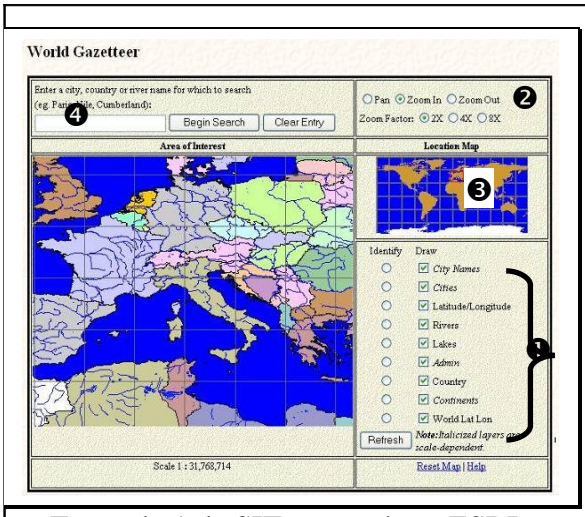
Il existe un très grand nombre de solutions cartographiques. Les principaux produits du marché sont représentés par les grands éditeurs de logiciels SIG [GEORE] :

- *ESRI* : leader du marché SIG, les logiciels d'Esri se déclinent sous deux aspects : l'aspect développeur d'applicatifs SIG avec la solution ArcInfo 8 qui fonctionne sous Windows NT, et l'aspect utilisateur du SIG avec les solutions ArcView.
- *GEOMEDIA d'INTERGRAPH*[22] : le logiciel Géomedia est dédié aux utilisateurs de SIG. Son atout réside essentiellement dans sa capacité à intégrer et lire directement la plupart des formats du marché.
- *MAPINFO*[23] : logiciel à l'origine de la démocratisation de l'utilisation du SIG sur PC, il est aujourd'hui très largement répandu.
- *GEOCONCEPT*[24] : leader des logiciels SIG français, Geoconcept possède toutes les fonctionnalités des SIG bureautiques avec un plus pour l'aspect saisie de l'information descriptive.

Quelques interfaces utilisateur de Systèmes d'Information Territoriale (SIT) sont données ci-dessous à titre d'exemple. Ces réalisations, ayant une thématique[25] proche de la nôtre sont

[22] Géomédia : http://www.intergraph.com/
[23] MapInfo : http://www.mapinfo.com
[24] Geoconcept : http://www.geoconcept.com/
[25] Thématique au sens de la représentation cartographique et les interactions que l'on peut observer.

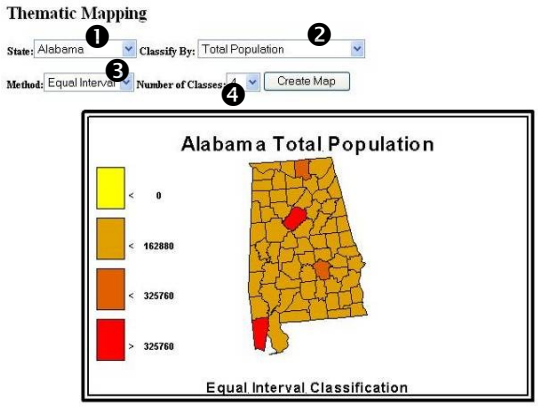
disponibles sur Internet. Elles permettent d'évaluer brièvement les interactions disponibles avec la carte et l'ergonomie de l'interface.



Exemple 1 de SIT proposé par ESRI.

L'exemple ci-contre, proposé par la société ESRI, montre la possibilité, courante dans un SIG, de représenter un ensemble de couches❶ qui se superposent (e.g. pays, rivières) sur une même carte.
Une autre possibilité fréquente est la fonction de zoom❷ et la notion de vue globale❸.
La recherche d'une ville ou d'un état est aussi possible en mode texte❹.

Principales critiques :
- la légende est peu significative,
- pas de possibilité de sélection de zones géographiques à la souris,
- une seule représentation du détail (ici le pays).

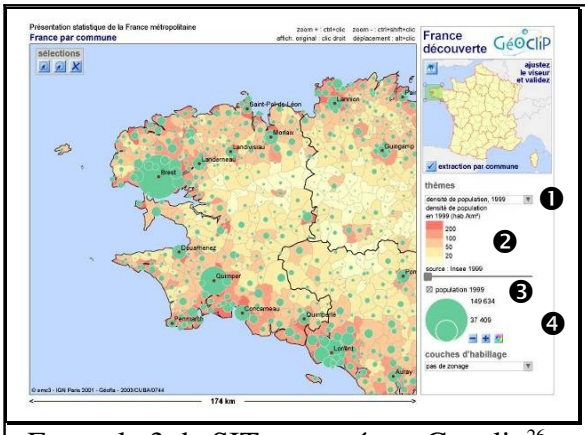


Exemple 2 de SIT proposé par ESRI.

Cet autre exemple est plus orienté thématique "socio-économique". On peut sélectionner :
- la zone sur laquelle porte l'étude (ici l'Alabama)❶,
- le thème choisi (la population totale)❷,
- le mode de répartition (intervalle équidistant)❸,
- le nombre de classes (4 classifications)❹.

Cet exemple se fond davantage dans la thématique de notre projet, *des critiques* peuvent cependant lui être adressées :
- pas d'échelle représentée,
- la méthode utilisée pour le calcul des 4 classes peut être optimisée (pas de zone représentée en jaune) d'où une "perte" d'information.

Et comme le cas précédent :
- pas de possibilité de sélection de zones géographiques à la souris et,
- une seule représentation du détail (e.g. échelle).



Exemple 3 de SIT proposé par Geoclip[26].

Le SIT ci-contre proposé par Géoclip marie habilement :
- une thématique représentée, ici la densité de population en 1999❶ pour chaque commune, par l'intermédiaire de 5 couleurs❷,
- le critère « population en 1999 »❸ est représenté par des cercles dits proportionnels de couleur verte,
- la possibilité éventuelle de juxtaposition de couches (couche d'habillage)❹.

Même si cet exemple significatif est le plus intéressant, il est critiquable sur :

	- le degré de détail pour la représentation qui reste toujours la commune et, - le nombre de classes fixé à 4.

La première réalisation ne présente que des fonctionnalités cartographiques de base (affichage), tandis que les deux autres utilisent nécessairement des fonctions de calcul plus élaborées (répartition de la population et cercles proportionnels).
Ces interfaces sont simples d'utilisation mais reposent, pour les deux dernières solutions, sur des critères attributaires (e.g. requête thématique). Nous avons vu précédemment (section II.1.5.2) que les requêtes ou opérations spatiales représentent une fonctionnalité essentielle pour les applications que l'on peut qualifier de SIG. Celles-ci sont néanmoins possibles pour, par exemple, afficher en couleur les communes traversées par une rivière. Mais ces requêtes font le plus souvent appel à des connaissances minimales en SQL, ce qui impose d'appréhender le modèle relationnel des données et les destinent tout naturellement à un public averti.

II.1.7 Conclusion : réponse à nos besoins

Les produits du marché proposent un large éventail de solutions, pas toujours disponibles via le Web, qui tentent de répondre aux problématiques spécifiques des domaines concernés (aménagement du territoire, gestion des transports, etc.). Cependant, l'interface utilisateur n'offre pas toujours l'ergonomie attendue et ne respecte pas toujours les fondements de la cartographie (visualisation d'une échelle…). Si l'on retrouve généralement les fonctionnalités classiques d'un SIG (zoom, déplacement, etc), les possibilités d'interactions plus spécifiques pour des traitements élaborés comme le choix du type de progression (arithmétique ou géométrique) n'est pas toujours proposé en standard.

D'autre part, l'analyse territoriale multiscalaire que nous devons étudier est basée sur l'hypothèse qu'il n'est pas possible d'évaluer la situation d'une unité territoriale donnée sans tenir compte de son contexte (unités voisines, pays d'appartenance…). En effet, d'un point de vue politique et dans une perspective sociale, les contrastes et les gradients présentent beaucoup plus d'intérêts que des valeurs absolues isolées. Certaines solutions de SIT offrent la possibilité d'analyser conjointement deux critères socio-économiques pour offrir à l'utilisateur la cartographie d'un phénomène (typiquement le ratio PIB par habitant). Cependant, ces déviations (e.g. comparaison) proposées, n'impliquent généralement que deux données attributaires et ne répondent que partiellement à notre problématique « multiscalaire ». Une réponse peu néanmoins être apportée, mais en appréhendant les nombreuses commandes SQL ainsi que le modèle utilisé, pour exprimer explicitement les différentes analyses attendues, ce qui ne correspond pas à notre volonté de diffusion à un large public.

Ainsi, à ce jour, aucun SIG commercial ne répond à notre problématique d'analyse multiscalaire proposée à travers un environnement cartographique convivial. Il est donc nécessaire de développer une application spécifique. Elle devra être capable d'intégrer les notions de contiguïté (e.g. analyse spatiale), d'analyse hiérarchique et de synthèse des déviations non disponibles dans les produits actuels. Cette interface de visualisation devra en outre utiliser les technologies Internet pour la rendre disponible via le Web aux nombreux acteurs, potentiellement intéressés par un produit libre (sous licence GPL).

[26] Géoclip : http://www.geoclip.net

II.2 Cartographie interactive sur Internet

II.2.1 Introduction

La conception de systèmes cartographiques performant et l'essor des moyens de communication lié à diverses technologies de l'information ont fait naître de nouveaux horizons comme la diffusion cartographique sur le Web [SOUSS01] qui semble vouée à un grand avenir. En effet, si l'engouement pour les logiciels SIG s'explique par une offre en adéquation avec les utilisateurs, la part de marché prévisionnelle des SIG intégrant les technologies Web, comme le prototype développé dans ce mémoire, ne cesse de s'accroître et devrait se situer à 67% de l'ensemble des SIG du marché en 2004 pour seulement 15% en 1999 [IDC04].

Le navigateur Internet devient ainsi l'interface standard pour accéder à un nombre croissant de services ou d'applications liés à la cartographie pour afficher des données et interagir avec elles. Les solutions engendrées par ces services sont généralement proposées dans la littérature selon les différents niveaux d'interactivité résultant des fonctionnalités offertes à l'utilisateur pour agir sur la carte. [GMMPV04] distingue trois niveaux :
Le premier niveau consiste à rendre une carte sélectionnable grâce aux seules possibilités offertes par le langage HTML (HyperText Markup Language) [HTML].
Les fonctionnalités de zoom, de déplacement ainsi que de présentation (légendes, affichage de couches, changement d'échelles, etc.) constituent le deuxième niveau regroupé sous la terminologie de Web-mapping. Tandis que les solutions qui tentent d'apporter toutes les fonctions habituelles des « 5A » font généralement référence aux SIG en ligne (e.g. le troisième niveau).

Dans cette partie, seules les techniques qui offrent des interactions (faibles ou évoluées) avec la carte via le navigateur sont proposées.

II.2.2 La diffusion cartographique : les stratégies possibles

La diffusion de l'information géographique sur le Web nécessite une architecture et des technologies adaptées afin de répondre à l'objectif d'un système Web (figure 22). Celui-ci se compose d'un navigateur Web ❶, d'un protocole de communication Internet ❷, d'un serveur Web ❸, de données, éventuellement distantes ❺ afin de produire les documents ❻ à fournir au client.

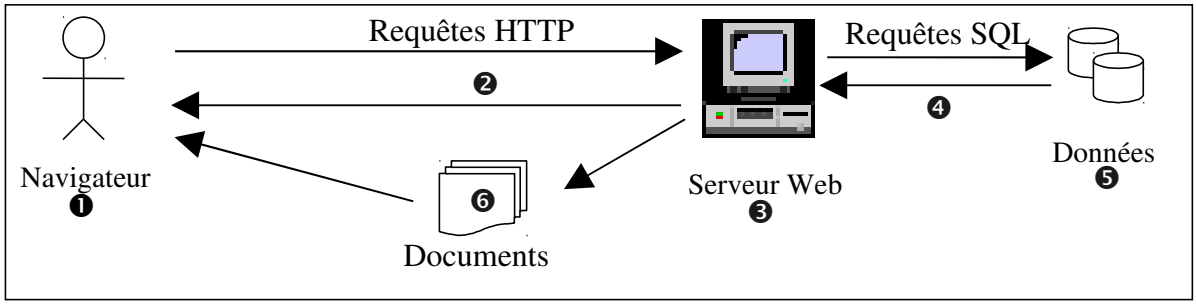


Figure 22 : Schéma de principe d'un système Web.

Son fonctionnement est le suivant : l'utilisateur ❶ se connecte via son navigateur au serveur Web distant ❸. En saisissant une URL, une requête HyperText Tranfer Protocole (HTTP) ❷, correspondant au protocole de transfert le plus courant, est envoyée au serveur Web. Celui-ci traite la demande en association avec les données ❺, le plus couramment stockées dans des bases

de données accessibles généralement avec des requêtes SQL ❹. Le document ❻, produit le plus souvent en langage HTML est fourni au client. Ce langage de présentation de données permet la consultation à partir de n'importe quelle plate-forme disposant d'un navigateur Web.

Cependant, si le navigateur répond bien à son objectif d'interface universelle pour la consultation de tous types de données (textes, images, graphiques, sons, animations) sur Internet. Il offre bien peu de possibilités pour satisfaire aux besoins cartographiques en terme de visualisation mais surtout d'interaction. L'application cartographique devra donc nécessairement accroître les fonctionnalités de base offertes pour au moins l'un des protagoniste principaux de la transaction (e.g. le client, le serveur ou les deux).

Deux stratégies fondamentales peuvent être ainsi retenues.
La première stratégie vise à étendre les fonctionnalités au niveau du serveur afin de permettre les interactions avec l'objet cartographique et sa consultation sur un navigateur Internet de base. A l'inverse, la deuxième stratégie consiste à rendre le poste client plus autonome.
[CONAL00] distingue trois types de clients parmi lesquels le client Web léger et client Web lourd. Le client Web léger utilise les seules fonctionnalités de base du navigateur (e.g. présentation). Le client Web lourd aura en charge des fonctionnalités de traitement (en plus de la présentation) grâce à l'utilisation de diverses technologies, à savoir l'utilisation de scripts et d'objets personnalisés comme des contrôles ActiveX et des applets Java du côté client.
Face à ces deux grandes stratégies qui visent à déporter de "l'intelligence" soit sur le client, soit sur le serveur, toutes les alternatives sont possibles et on cherchera notamment à mieux répartir la charge de travail entre client et serveur pour accroître la performance du système.

Les différentes solutions proposées [GMMPV04] dans la suite du document répondent toutes à notre problématique de diffusion de cartographie interactive avec cependant des degrés d'interactions différents. Elles sont regroupées par grandes stratégies, si l'on souhaite étendre les possibilités du client ou du serveur. En prenant en compte les alternatives possibles qui se révèlent la forme la plus courante d'utilisation. L'objet cartographique (e.g. image raster ou image vectorielle) sera pris en considération tout comme les « manières » de le diffuser (dynamique ou statique).

II.2.3 Stratégies côté client

Les stratégies client visent à exploiter les possibilités du navigateur Web de base comme la première solution proposée ci-après, jusqu'à utiliser de réels programmes afin d'entreprendre des manipulations de données et/ou des analyses complexes sur la machine du client.

II.2.3.1 Un client léger pour une carte dite « morte »

La diffusion de cartes sur Internet la plus facile à mettre en œuvre est de stocker les documents contenant les images directement sur le serveur (pas de création « à la volée »).

Une carte peut être créée manuellement ou plus simplement produite à l'aide d'un logiciel SIG de bureautique en exportant le résultat cartographique au format raster. Cette image sera alors insérée dans une page HTML, au même titre que n'importe qu'elle autre illustration afin de la rendre visible à l'utilisateur par le biais de son navigateur.

Les formats d'images [FORMA] qui peuvent être nativement affichés par les navigateurs sont le GIF (Graphics Interchange Format), le JPEG (Joint Photographic Expert Group) et le PNG (Portable Network Graphic). Le format GIF est limité à 256 couleurs mais restitue une image

sans perte d'information. Le format JPEG affiche jusqu'a 16,7 millions de couleurs mais connaît une dégradation de l'image engendrée par ses possibilités de compression. Le format PNG, représente un standard libre de tout droit émanant du W3C (World Wide Web Consortium) [W3C], l'organe régulateur du Web pour la définition des nouveaux standards. Il cumule les avantages du JPEG pour les couleurs et utilise un mode de compression sans perte d'information.

Ce type de carte offre bien peu de possibilités pour satisfaire aux besoins cartographiques d'interaction d'où son appellation de carte morte ou de carte à lire. Cependant, les possibilités du langage HTML permettent d'intégrer au sein d'une image, des zones réactives (e.g. des images raster) afin d'autoriser une navigation liée à la zone sélectionnée. Tous les formats raster cités peuvent être incorporés à l'aide de l'élément HTML « img » et offrir des renvois grâce à des liens hypertextes afin d'obtenir, par exemple, un zoom pré-calculé de la zone sélectionnée.

Si la diffusion de cartes statiques au format image offre des possibilités restreintes, elle peut cependant s'avérer suffisante. Cette méthode est rapide à mettre en œuvre et peu onéreuse car elle ne nécessite qu'une architecture simplifiée. Elle offre aussi la meilleure garantie d'une restitution conforme aux attentes grâce à l'utilisation des formats standard d'Internet (e.g. les formats raster), même si ces formats souffrent rapidement de dégradation de rendu.

II.2.3.2 Des plug-in pour des solutions vectorielles

Les formats vectoriels contiennent une description des entités géométriques à représenter comme les formes géométriques élémentaires (lignes, cercles…), les surfaces plus complexes (e.g. polygones) en précisant diverses indications telles que les couleurs de remplissage, etc.
Cette richesse d'information bien supérieure à celle d'une image est exploitable grâce à des logiciels d'application complémentaires, les plug-in (ou Plugiciel) téléchargés (une fois) sur le poste client et résidant sur celui-ci. Le plug-in en s'associant avec le navigateur Web, permet de visualiser d'autres formats de fichier mais surtout d'accroître les fonctionnalités du navigateur et, par conséquent, les interactions avec l'objet cartographique.

Comme chaque format utilise un modèle vectoriel qui lui est propre, il sera toujours associé au plug-in (généralement celui fourni par le concepteur) pour l'exploiter. Ainsi, si le taux de diffusion des différents plug-in ne doit pas être déterminant, il peut néanmoins influencer le choix d'une solution et donc de son format. Les résultats d'un relevé trimestriel, couvrant les Etats Unis, organisé par un institut indépendant NPD-Mediametrix[27] (Juin 2003) montrent les tendances actuelles (figure 23).

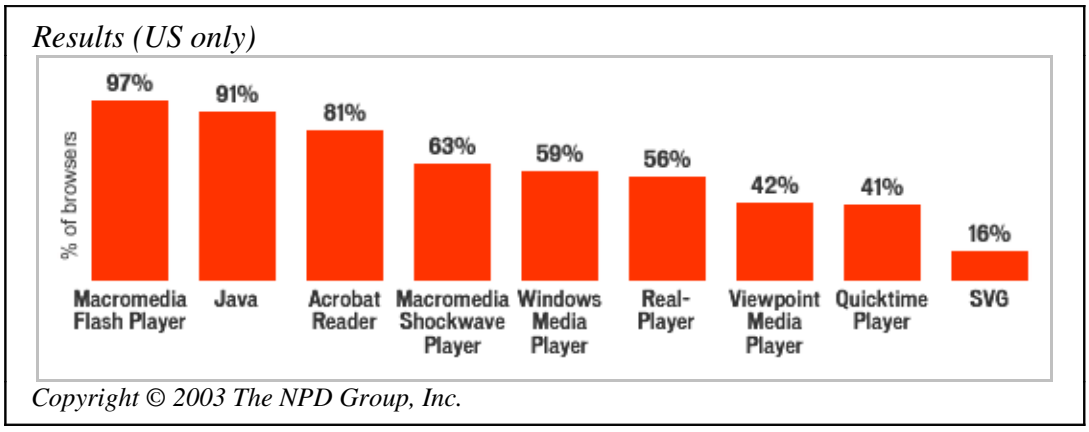


Figure 23 : Statistiques d'équipement en visionneuses Internet.

[27] NPD-Mediametrix : www.npd.com

II.2.3.2.1 Les formats vectoriels

Une alternative à la solution précédente (e.g. image morte) consiste à exporter une carte dans un format vectoriel. Sauvegardée sur le serveur, elle sera accessible par le client qui pourra directement bénéficier des fonctionnalités fournies par le seul plug-in, tout en profitant d'une qualité graphique irréprochable, que ce soit à l'écran comme à l'impression.
Les différents formats retenus pour être présentés offrent chacun des particularités remarquables. Le format PDF[28] (Portable Document Format) reste une solution largement répandue. La solution Flash[29] avec son format SWF (Flash Shockwave) bénéficie du plus grand taux de diffusion de son plug-in (figure 23). La norme SVG (Scalable Vector Graphics) [SVG] peut apparaître comme une solution d'avenir et sera développée dans la section II.3. Tandis que le format GML (non représenté dans la figure 23) tente plus particulièrement de répondre à des normes de stockage de la donnée géographique.

II.2.3.2.1.1 Le format PDF

Le format PDF est la propriété de la société Adobe[30]. Il s'appuie sur la syntaxe EPS et tant à remplacer progressivement le Postscript. Associé à un plug-in comme Acrobat Reader, il permet de représenter tout type de document et *a posteriori* une carte insérée dans celui-ci. Cette solution permet de gérer une carte au format raster mais reste plus spécifique au format vecteur. Le format est multi plates-formes, robuste et respecte l'apparence du document initial, à l'écran comme à l'impression. La lecture, la copie ou l'impression peuvent être protégés par un mot de passe.
Si cette solution est propriétaire, on peut cependant trouver des alternatives grâce au produit PDFcreator de la communauté Sourceforge[31], spécialisée en logiciels open source. Il permet de convertir très facilement des fichiers imprimables en fichiers PDF (via le format Postscript).
Grâce à son plug-in gratuit et indépendant du navigateur, il reste une forme largement utilisée pour proposer des cartes en téléchargement comme la diffusion de plans d'accès.

II.2.3.2.1.2 Le format SWF

Flash est un logiciel de la société Macromédia[32] qui utilise des fichiers (au format SWF) destinés à être intégré dans des pages HTML au même titre que des images GIF ou JPEG. Cette alternative représente actuellement le format vectoriel permettant l'interactivité le plus diffusé (figure 23). Dédié au départ aux animations graphiques des sites Web, le monde de la cartographie s'est emparé de ce mode de diffusion pour publier des cartes. L'utilisation de fichiers au format SWF nécessite l'ajout d'un plug-in gratuit (Flash Player) souvent déjà installé sur les navigateurs. Cette solution est aussi la plus aboutie pour représenter des vecteurs dans le domaine de la cartographie. Les éléments graphiques peuvent être importés depuis Macromedia *FreeHand*, Adobe *Illustrator* ou *CorelDraw* et/ou édités avec Macromedia *Flash*.
Le format SWF est propriétaire mais bénéficie d'un format binaire très compressé et apporte des fonctionnalités intéressantes sur le Web, comme le fait de rendre visible une page même si toutes les données ne sont pas encore chargées.

II.2.3.2.1.3 Le format SVG

Le format SVG est un format de stockage et d'échange de données graphiques en 2D. Il est plus récent que les deux formats déjà présentés (première diffusion en 2001), ce qui explique une diffusion plus restreinte (figure 23).

[28] PDF : http://www.adobe.fr/products/acrobat/adobepdf.html
[29] Flash : www.macromedia.com/software/flash/
[30] Adobe : http://www.adobe.com
[31] Sourceforge : http://sourceforge.net/
[32] Macromedia : http://www.macromedia.com

Il s'agit d'un standard ouvert spécifié par le W3C[33] (World Wide Web Consortium). Issu de la norme XML (Extensible Markup Language) [XML], il bénéficie de ses spécifications et des nombreux outils capables de structurer et manipuler les informations. SVG décline un format compressible (extension SVGZ). On peut le coupler avec des données raster. Un document SVG sera directement visible grâce à un plug-in, comme le SVG-viewer que propose la société Adobe gratuitement.
Face aux solutions du langage Flash, qui représente un standard de fait, SVG semble voué à un bel avenir si l'on prend en considération les spécifications respectives des deux formats [FLSVG].

II.2.3.2.1.4 Le format GML

Le format GML, comme la norme SVG, est issu de la norme XML mais spécialisé dans le domaine de l'information géographique. Il est destiné au transport et au stockage des données spatiales (géométrie) et des données attributaires. Mais surtout, grâce à l'Open GIS Consortium[34] dont il est issu, il tente de répondre à l'interopérabilité entre les différentes solutions (e.g. formats) proposées.
Cette norme offre un cadre d'échange neutre (e.g. non propriétaire) des données géographiques, adapté aux volumes de données petits à moyens. L'utilisation d'outils applicables à XML comme le principe de filtre XSL (Extensible Stylesheet Language) permet de limiter les données transmises aux besoins et contraintes spécifiques, tandis que le principe de transformation XSLT (XSL Transformations) permet de fournir une représentation visuelle des données adaptée au support de consultation.

II.2.3.2.1.5 Conclusion

L'usage de ces formats avec l'adjonction exclusif de leurs plug-in respectifs apporte peu d'interactivité avec la carte si ce n'est quelques fonctionnalités comme le zoom ou éventuellement des rotations. Cependant, ces solutions apportent une grande qualité d'images sans dégradation. Si elles restent un bon standard de diffusion, la richesse du format qui s'apparente à un véritable « objet cartographique » laisse entrevoir de nombreuses possibilités d'interactions.

II.2.3.2.2 Des programmes pour augmenter l'interaction

Afin d'augmenter les fonctionnalités natives des plug-in, tous ces formats font généralement appel à des programmes clients interprétés tels des scripts. Les plus connus sont le VBScript mais limités à un environnement Microsoft et javascript qui pourra exploiter des solutions flash ou le format SVG.
Les codes de ces scripts sont insérés dans une page HTML puis chargés par le visiteur par l'intermédiaire du navigateur pour être exécuté chez le client suite à un événement. Les événements sont des actions de l'utilisateur, qui vont pouvoir donner lieu à une interactivité. Si le « clic » de souris est déjà directement géré avec le langage HTML, d'autres méthodes ou fonctions peuvent être associées à des événements tels que le passage de la souris au-dessus d'une zone, le changement d'une valeur etc.

Les scripts dédiés au format propriétaire représentent une alternative aux scripts « généralistes » déjà cités. Ainsi le langage Acrobat JavaScript d'Adobe permet d'intégrer des scripts ECMAScript dans les documents PDF, interprétés grâce au module externe ECMAScript, tandis

[33] W3C : http://www.w3c.org/
[34] OGC : http://opengis.net/gml

que le lecteur Flash utilise son langage ActionScript pour apporter de véritables fonctionnalités d'animations et d'effets spéciaux.

Si l'utilisation des langages de scripts peut déjà accomplir de nombreuses fonctions d'interactions avec la carte, celles-ci peuvent encore être étendues par le biais de langages plus performants tels que Java (présenté dans la partie suivante) ou tout autre langage capable d'exploiter la richesse des formats vectoriels.

II.2.3.3 Les applets Java et les contrôles ActiveX

Une applet Java ou un composant ActiveX se présente sous la forme d'un programme exécutable, inséré dans une page HTML. Le navigateur Web « compatible » reconnaît l'application à traiter, la charge et l'exécute sur la machine cliente (e.g. en local).
Contrairement à la technique des plug-in, téléchargés une fois et résidant à titre permanent pour le navigateur, une applet Java ou un composant ActiveX est téléchargé depuis le serveur Web sur le poste client au moment de la connexion. Cette technique n'impose donc aucune installation manuelle de la part de l'utilisateur contrairement au plug-in. Par contre, ce type de solution est parfois qualifié de « logiciels jetables » car l'applet ou l'ActiveX doit être téléchargé à chaque fois que l'internaute accède au document Web contenant l'application. D'autre part, les composants peuvent être parfois lourds à télécharger pour offrir de nombreuses fonctionnalités (e.g. client lourd).

Ces technologies fonctionnent suivant une architecture de type client car c'est le client qui exécute la majorité des traitements. Cependant, plusieurs nuances peuvent exister suivant le degré d'implication du serveur. En effet, la totalité de l'application et des données peut être téléchargée lors d'une seule requête utilisateur. Tandis que la solution plus orientée client serveur (décrite dans la suite du document) nécessite (généralement) plusieurs requêtes utilisateurs.

La figure ci-dessous (figure 24) présente le principe de fonctionnement d'une applet Java indépendante.

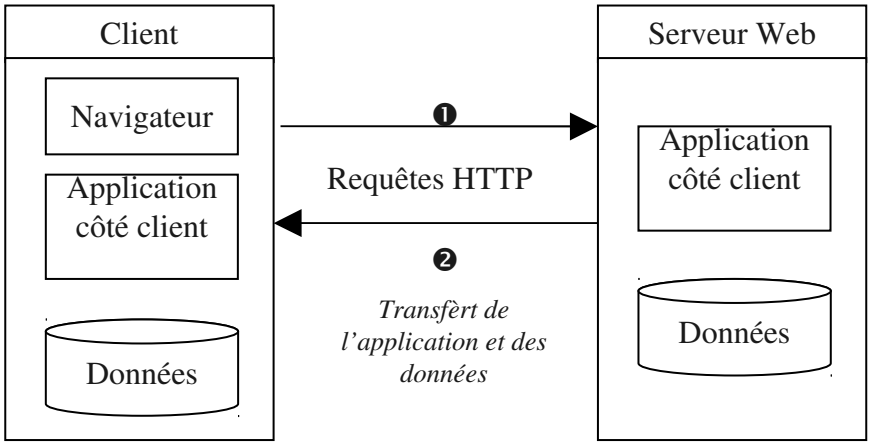


Figure 24 : Principe de fonctionnement d'une applet Java.

Le navigateur du client reconnaît une applet à télécharger dans sa page HTML lors d'une (seule) requête utilisateur ❶. Le serveur envoie au client l'application (côté client) avec toutes les données qu'il sera en mesure de traiter ❷. Ces données sont alors stockées dans le cache du client en même temps que l'applet pour être visualisées.

Cette stratégie est qualifiée d'« indépendante » et devient analogue à celle d'une application monoposte. En effet, le client sera ainsi en mesure d'effectuer toutes les opérations de visualisation ou d'analyse prévues par les concepteurs, sans autre intervention du serveur. Toute l'interactivité étant gérée à partir du poste utilisateur, le client sera qualifié de client lourd.

Ce type d'application permet de gérer des formats vecteur et/ou raster et l'interactivité ne se limite qu'aux seules fonctionnalités prévues par le concepteur. On retrouve ainsi couramment les fonctionnalités qui s'appliquent à la cartographie comme le zoom, les déplacements, l'affichage de données attributaires correspondant à une sélection, la gestion des couches, etc. De plus, comme il s'agit de véritables langages compilés contrairement aux langages de scripts interprétés, de véritables interfaces graphiques peuvent être implémentées pour faciliter les interactions avec l'utilisateur.

Les composants ActiveX ou applets Java sont donc destinés à accroître les fonctionnalités du navigateur Web. Ils peuvent être utilisés dans une infrastructure matérielle très légère. En effet, un serveur non spécialisé peut héberger l'application cartographique « insérée » dans un document HTML standard. Si les deux produits fournissent des fonctionnalités comparables, l'utilisation d'un contrôle ActiveX se limite à des environnements Microsoft. Même si son navigateur Internet Explorer reste de loin le plus diffusé, la solution des applets java dans un environnement multi plates-formes (C, UNIX, MacIntosh …) reste le choix le plus répandu. Ce choix peut aussi s'expliquer par la qualité des interfaces utilisateur que l'on peut réaliser grâce à la grande bibliothèque de composants que Java fournit.

II.2.4 Stratégies côté serveur

La stratégie serveur vise à accroître les fonctionnalités du serveur afin d'offrir des services de cartographie interactive avec l'utilisateur. Ces services conduisent généralement à la production de pages dynamiques (e.g. carte créée à la volée) en fonction de la demande de l'utilisateur ou de son environnement (langue…). Ils apportent aussi une interaction directe avec la carte produite grâce à des échanges entre client et serveur avec l'utilisation de différentes technologies.

II.2.4.1 Une carte interactive créée dynamiquement

La solution la plus « robuste » pour diffuser une carte sur Internet est de générer des images côté serveur pour les publier ensuite sur les navigateur Web.
Dans cette stratégie de type serveur (figure 25), l'utilisateur envoie une demande à un serveur Web (e.g. une requête) ❶. Cette demande est généralement provoquée par l'intermédiaire d'un formulaire HTML (contenant du code Javascript, des applets, etc.) qui permet de recueillir les données de l'utilisateur (e.g. la carte à produire). Le serveur Web identifie l'environnement d'exécution (technologie employée), charge puis exécute le script où programme serveur ❷ en relation avec le serveur d'application qui peut être spécialisé dans le traitement géographique (serveur SIG). Cette exécution entraîne naturellement des connexions avec une base de données ❸ qui fournit les données nécessaires pour le traitement en cours. Le serveur d'application fournit ensuite le résultat (e.g. image raster créée à la volée) au serveur Web ❹ qui se chargera de l'intégrer dans la réponse (flot de sortie) à destination du navigateur ❺. Le client peut alors visualiser sa carte grâce à son navigateur standard.

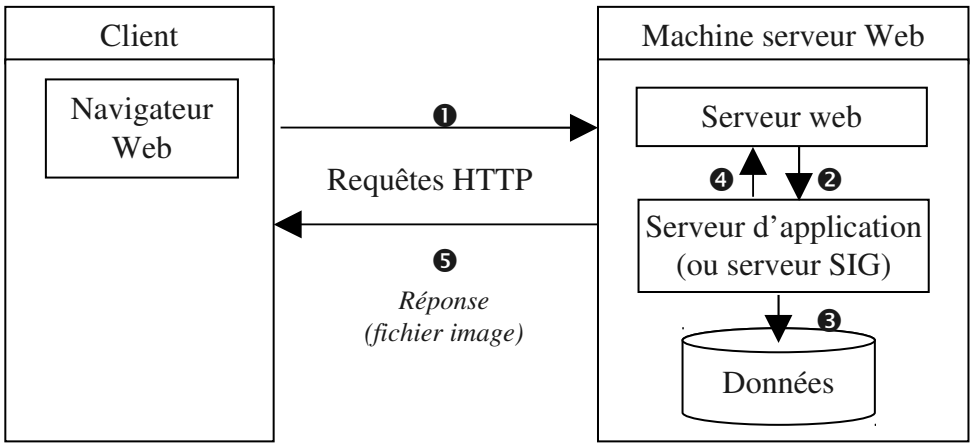


Figure 25 : Architecture serveur [GLEYZ01].

Cette technique est fiable car elle garantit que le résultat cartographique sera visible immédiatement sur tous les navigateurs (e.g. client léger) sachant lire les formats raster les plus courants comme les formats GIF, JPEG ou PNG.

Dans ce type de configuration, l'interactivité peut revêtir la forme d'une nouvelle production de carte, par exemple afin d'afficher une couche différente ou changer l'échelle de représentation, etc., tandis que l'interactivité avec l'objet cartographique (e.g. contenu dans la carte) nécessitera l'utilisation d'images (raster) pourvues de zones réactives avec une petite application (généralement une applet java ou un javascript). Lorsque une zone réactive est sollicitée, une requête utilisateur est envoyée au serveur Web. Cette requête précise les coordonnées écran de la souris à l'origine de l'événement (« clic » droit, gauche, sélection …) ainsi que les actions à effectuer grâce à l'applicatif du client. L'environnement serveur traduit alors les informations en coordonnées réelles et en actions à gérer.

II.2.4.2 Des technologies pour des pages dynamiques

Même si les documents à diffuser vers le client peuvent être directement stockés (au format HTML, image ou autre) dans sa version définitive sur le serveur, la solution qui consiste à générer « à la volée » ces mêmes documents retient notre attention (e.g. source d'interactions).
Le document à obtenir du côté client sera généré lors du traitement d'une requête utilisateur grâce au serveur d'application qui fournit des services (programmes) déclenchés par des technologies dites serveur. Les technologies les plus courantes afin de générer dynamiquement ces pages HTML ou plus généralement des solutions pour le client sont les scripts CGI (Common Gateway Interface), les programmes PHP (Personal Home Page), les programmes ASP (Active Server Pages de Microsoft) et les solutions Java, les servlets et les JSP (Java ServerPages).

- CGI est un protocole qui est utilisé par le client pour envoyer des données au serveur Web (sous la forme d'un formulaire par exemple) qui les transmet à son tour à une application externe résidant sur la même machine afin de construire des pages HTML dynamique qui seront renvoyées au client. Même si ce type de passerelle est grosse consommatrice de ressources (un accès déclenche un processus autonome sur le serveur), les programmes CGI sont les plus courants et peuvent être développés avec de nombreux langages de programmation notamment Perl et C.

- PHP est un langage en open source utilisé dans des applications Web pour écrire des scripts HTML. L'essentiel de sa syntaxe est emprunté aux langages C, Java et Perl. Lorsqu'un client effectue une requête sur une page développée en langage PHP, le script

est préalablement évalué et interprété côté serveur. Cette interprétation aboutit généralement en la création d'une page HTML destinée au client. Le langage PHP est simple à utiliser et permet d'inclure son script directement au sein d'une page HTML.

- ASP est une technologie de Microsoft pour la création dynamique de pages Web. Le langage de base pour développer un fichier ASP est VBScript. Cette technologie pourra être combinée avec les deux langages de balisage, HTML et XML, avant d'acheminer le résultat vers le poste client. Si elle concurrence les programmations CGI et PHP, cette solution répandue nécessite que le serveur accepte les programmes ASP comme Microsoft Internet Information Server (IIS), la solution de Microsoft.

- Un Servlet est un programme Java qui correspond à une applet Java exécutée dynamiquement sur le serveur Web. Lorsqu'un client envoie une requête au serveur, ce dernier transmet au servlet les informations relatives à la requête. Par la suite, le servlet crée une réponse que le serveur renvoie au client. Lors de la création de la réponse, le servlet peut utiliser toutes les fonctions du langage Java ou communiquer avec des ressources externes. De plus les servlets bénéficient d'une solution multi plates-formes car elles sont indépendantes du serveur Web en s'exécutant dans un moteur de servlet utilisé pour établir le lien entre le servlet et le serveur Web.

- La « JSP » est une page qui contient des balises HTML classiques (ou autres éléments d'un langage de balisage) et des éléments JSP permettant d'insérer du contenu dynamique. Ces éléments propres au JSP sont appelés des "scripts" généralement écrits en langage Java mais aussi en langages JScript, Perl ou VBScript. L'intérêt principal de ce mécanisme par rapport aux servlets provient de la séparation du codage HTML de la logique applicative fournie par Java. Le principe d'utilisation est semblable à la technologie ASP de Microsoft avec normalement plus de portabilité car ils permettent d'inclure directement du code Java dans une page HTML.

II.2.4.3 Solution serveur cartographique

Les serveurs cartographiques étendent les limites des serveurs classiques (e.g. mutualisés). Leurs spécialisations permettent de s'adapter aux contraintes de la cartographie dynamique et d'offrir de nombreuses interactions avec l'utilisateur. Ils sont généralement employés lorsque les données sont fortement évolutives, quand la couverture spatiale est étendue et que différentes échelles de visualisation sont demandées [GMMPV04]. Ces solutions intègrent notamment un environnement de production et une palette d'outils d'administration et de création de services pour tenter de parvenir aux mêmes gammes de fonctionnalités que l'on peut retrouver sur un SIG classique. Les formats utilisés répondent le plus souvent à l'OpenGIS à des fins d'interopérabilité pour communiquer entre les différentes bases. Des formats plus standards alliés à des interactions plus riches et spécifiques en fonction des besoins des clients constituent de véritables atouts pour des solutions Web.

L'architecture ci-dessous montre une solution pour un serveur cartographique dont les données attributaires et géographiques sont gérées séparément (figure 26) [GMMPV04].

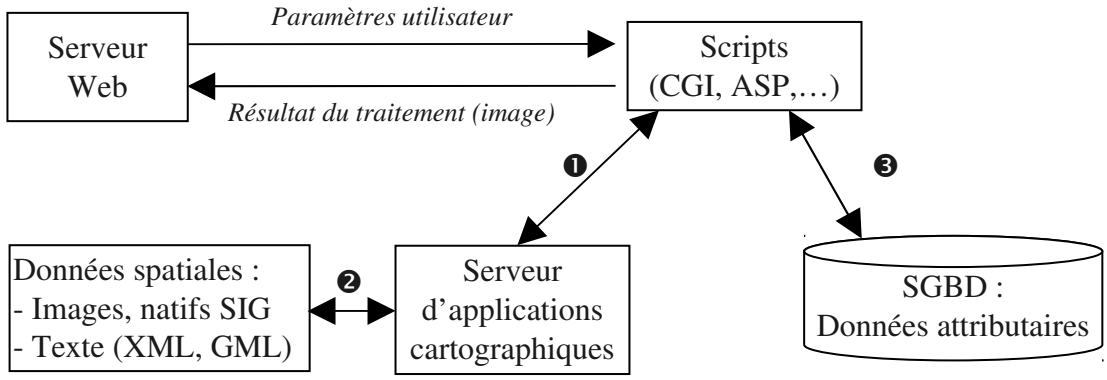


Figure n° 26 : Architecture pour une solution « serveur cartographique »[35].

Le fonctionnement sera étudié avec l'exemple d'une zone réactive de la carte sélectionnée à la souris chez le client permettant de retrouver le nom de la zone.

Le client fournit au serveur Web les coordonnées du curseur de la souris lors du « clic ». En fonction des paramètres utilisateur, les scripts permettent de transmettre les requêtes aux éléments architecturaux concernés. L'identification de l'objet concerné (sa géométrie) utilise les services du serveur cartographique ❶. Après avoir traduit les coordonnées écran (e.g. utilisateur) en coordonnées système, il interroge la base de données géographique ❷ pour connaître l'objet géométrique qui « englobe » le « clic » de souris. Au besoin, il crée à la volée les formes pour tester cette appartenance. L'objet géométrique identifié, il suffira de connaître sa correspondance (relation) avec l'objet contenant les données attributaires. La base de données attributaires sera alors interrogée ❸ sur le critère de recherche concerné afin de restituer via le serveur Web le nom de la zone sélectionnée au client.

II.2.5 Stratégies mixtes Client-Serveur

La stratégie mixte peut être vue comme un « mixage » entre les deux stratégies proposées précédemment. Elle tente de répondre au mieux aux exigences de la cartographie interactive en cumulant les avantages des deux situations.

Le produit Alov[36] représente une exemple type de cette solution client-serveur. Les fonctionnalités du client sont basées sur une applet Java tandis que le serveur intègre une technologie à base de servlets Java. L'échange des données entre le serveur et le client est plus efficace grâce à un téléchargement progressif des données (réduction du trafic). Les données spatiales peuvent se situer dans une base des données qui répond aux spécifications de l'OpenGIS.

L'architecture et sa stratégie client-serveur semble, en principe, la mieux disposée à fournir les services d'interaction avec la carte. La carte peut, en effet, être produite par un serveur (cartographique ou non). Le client utilisera les fonctionnalités d'interaction prévues par les concepteurs grâce à l'adjonction de composants logiciels plus ou moins lourds. Ces composants peuvent apporter de véritables interfaces cartographiques ou, plus simplement, la possibilité pour le serveur de reconnaître l'action à traiter. Le serveur grâce aux technologies serveurs est capable de répondre favorablement à la requête du client. La prochaine intervention du client peut avoir lieu, etc.

[35] Les données spatiales et attributaires sont gérées séparémment.
[36] Alov : http://alov.org/index_fr.html

Dans ce type de stratégie, la répartition des traitements à effectuer entre client et serveur est primordiale. Les traitements lourds, comme la création de la carte, peuvent être traités entièrement sur le serveur afin d'alléger le client (solution serveur). Tandis que le client pourra se charger de fonctionnalités plus ou moins évoluées comme les déplacements, les zooms etc. sans nécessairement faire appel à une nouvelle production de carte par le serveur (optimisation de la bande passante). Une bonne répartition des traitements ne trouve pas de réponse unique car toutes les combinaisons sont possibles suivant les désirs que l'on souhaite privilégier (temps de chargement, vitesse d'exécution d'une analyse...).

II.2.6 Conclusion - Synthèse

La cartographie interactive sur Internet regroupe un large éventail de solutions. Déjà, les seules possibilités du langage HTML permettent de reconnaître un événement (e.g. le « clic » de souris) et renvoyer une autre image (préalablement construite) grâce aux liens hypertextes provoquant ainsi une interactivité. Pourtant, si cette solution est facile à mettre en œuvre, elle ne peut prétendre aux fonctionnalités attendues d'une véritable cartographie interactive. On cherche toujours à étendre les fonctionnalités d'interaction avec la carte sur le poste client en proposant des solutions que l'on peut regrouper selon deux grands types de stratégie (côté client ou côté serveur).

Les stratégies client (figure 27) tentent de rendre le poste du client plus autonome. Les technologies envisagées peuvent prendre la forme du plug-in. Celui-ci permettra d'accéder au format vectoriel plus riche graphiquement qu'un format raster et d'offrir un certain nombre de fonctionnalités de base prévues par le plug-in (propriétaire ou non). Pourtant, l'usage exclusif du plug-in n'offre généralement que des cartes que l'on pourra qualifier de faiblement interactives (sauf pour la solution Flash). C'est en exploitant toute la richesse conceptuelle du format vectoriel que les solutions deviennent performantes. On utilisera au choix des langages interprétés (e.g. scripts) ou, plus performants comme les langages compilés Java ou ActiveX pour réaliser de véritables applications cartographiques interactives.

<table>
<tr><th colspan="5">Stratégies « côté client »</th></tr>
<tr><th colspan="5">Applicatifs (côté client)</th></tr>
<tr><th colspan="3">Avantages</th><th colspan="2">Inconvénients</th></tr>
<tr><td colspan="3">Format vecteur généralement employé
Meilleure qualité d'image
Interface graphique évoluée</td><td colspan="2">Complexe à développer
Exige un logiciel additionnel
Temps de chargement parfois long
Aucun respect des normes (formats)
Incompatibilité des navigateurs</td></tr>
<tr><th colspan="5">Technologies (côté client)</th></tr>
<tr><th colspan="2"></th><th>Principes</th><th>Avantages</th><th>Inconvénients</th></tr>
<tr><td rowspan="4">Formats pour les plug-in</td><td>PDF</td><td rowspan="4">Logiciel à télécharger une fois (hors maj) et complémentaire au navigateur pour lire du format vectoriel</td><td>Fiable et répandu</td><td>Limité à la diffusion</td></tr>
<tr><td>SWF</td><td>Puissant et très répandu</td><td>Solution propriétaire</td></tr>
<tr><td>SVG</td><td>Solution performante issue du W3C</td><td rowspan="2">Plug-in peu répandu</td></tr>
<tr><td>GML</td><td>Interopérabilité</td></tr>
<tr><td colspan="2">Applets</td><td>Programme Java chargé (à</td><td>Multi plates-</td><td>Lenteur au</td></tr>
</table>

		chaque première connexion) et exécuté par le navigateur	formes Interfaces utilisateur riches	chargement selon la taille de l'applet
ActiveX		Composant logiciel qui respecte le modèle COM (Component Object Model) chargé (à chaque première connexion) et exécuté par le navigateur Microsoft	Réutilisation des composants, interfaçage avec les logiciels Microsoft	Technologie propriétaire utilisable seulement dans l'environnement Windows
Script	**JavaScript**	Langages interprétés. Le code est inséré dans une page HTML puis chargé par le navigateur et exécuté	Très documenté (bibliothèques)	Lenteur
	VBscript			Propriétaire (Microsoft)

Figure 27 : Stratégies « côté client ».

Les stratégies serveurs (figure 28) visent à rendre possibles les interactions sur un navigateur Internet de base (e.g. client léger). Les solutions induites sont le plus souvent basées sur l'envoi d'images au format raster ainsi qu'un applicatif pouvant gérer les évènements produits par le client. L'envoi de format vectoriel n'est cependant pas exclu. Dans toutes les configurations, que ce soit en utilisant un serveur d'application standard ou spécialisé (e.g. cartographique) la production de cartes dynamiques sera effectué (le plus généralement) grâce aux mêmes technologies serveur décrites dans ce paragraphe.

<table>
<tr><th colspan="5">Stratégies « côté serveur »</th></tr>
<tr><th colspan="5">Applicatifs (côté serveur)</th></tr>
<tr><th colspan="3">Avantages</th><th colspan="2">Inconvénients</th></tr>
<tr><td colspan="3">Plus simple à développer
Plus facile à déployer
Plus facile à maintenir
Adhère aux normes d'Internet
Utilise un navigateur standard</td><td colspan="2">Interface utilisateur simpliste
Qualité graphique faible</td></tr>
<tr><th colspan="5">Technologies (côté serveur)</th></tr>
<tr><th colspan="2"></th><th>Principes</th><th>Avantages</th><th>Inconvénients</th></tr>
<tr><td colspan="2">CGI</td><td>Un processus par requête est exécuté sur le serveur et renvoie une page HTML au client</td><td>Universel et peut être écrit dans de nombreux langages.</td><td>Gros consommateur de ressources</td></tr>
<tr><td rowspan="3">Script</td><td>PHP</td><td rowspan="3">Script intégré dans la page HTML et interprété par le serveur au moment du chargement de la page</td><td rowspan="3">Facile à développer et clair</td><td rowspan="3">Langage propre au serveur Web</td></tr>
<tr><td>ASP</td></tr>
<tr><td>JSP</td></tr>
<tr><td colspan="2">Servlets</td><td>Code Java exécuté sur le serveur et renvoi du HTML</td><td>Portable quel que soit le navigateur et la machine client</td><td>Limité à HTML pour la présentation des résultats</td></tr>
</table>

Figure 28 : Stratégies « côté serveur »

Les deux types de stratégie peuvent aussi être combinés pour optimiser les performances du système, comme la stratégie client-serveur. Toutes les options sont généralement permises si on exclut certaines solutions propriétaires. On tentera généralement de cumuler les avantages des deux stratégies pour mieux répartir la charge de travail entre le client et le serveur. Il en sera de même pour la bande passante et le transfert des données. Toutes ces réflexions devant répondre au mieux aux besoins de plus en plus spécifiques des clients en offrant une application conviviale et performante.

II.3 Technologies SVG et Java2D

II.3.1 Introduction

Dans le contexte de la cartographie, le choix de l'« objet » permettant de faire du dessin est essentiel. On se propose d'étudier deux grands standards parmi les produits ouverts. C'est-à-dire le langage SVG et l'API (Application Programming Interface) Java2D de Java. Ces deux « langages » permettent d'explorer les possibilités du dessin vectoriel en deux dimensions. Ils ont des philosophies suffisamment proches pour être comparés.
Il est bien sûr impossible de faire un état des lieux exhaustif de chaque produit, ce qui serait en outre fastidieux. C'est pourquoi, une brève description des deux langages est d'abord proposée. Puis, on dessinera une même figure géométrique avec les deux technologies. Ce qui a pour but d'introduire rapidement les mécanismes généraux des produits respectifs. Par la suite, les deux produits sont comparés simultanément sur diverses fonctionnalités qui peuvent avoir un attrait pour la cartographie.

II.3.2 La norme SVG

***S**calable **V**ector **G**raphics* est un « langage » reconnu par le W3C pour la description de graphiques 2D. Il a été inventé en 1998 par un groupe de travail (comprenant Microsoft, Autodesk, Adobe, IBM, Sun, Netscape, Xerox, Apple, Corel, HP, ILOG...) pour répondre à un besoin de graphiques légers, dynamiques et interactifs. C'est depuis l'apparition du plug-in d'Adobe en 2000 qu'il se place en concurrence avec la solution Flash.
Il fait partie de la famille des langages XML, ce qui lui permet de bénéficier de ses nombreux atouts.

II.3.2.1 SVG issu du langage XML

*e**X**xtensible **M**arkup **L**anguage* est un « langage » de description et d'échanges de documents structurés, centré sur les données. XML ne doit pas être vu comme un langage de programmation[37] mais comme un métalangage. Autrement dit, XML est un langage capable d'en décrire d'autres, SVG en fait partie.
L'un des principaux avantages d'utiliser XML comme source de données est que la présentation de ces données est séparée de leur structure contrairement au langage HTML. Les données XML définissent la structure et le contexte, alors qu'une feuille de style est appliquée pour définir la présentation. Un même document pourra être affiché différemment (e.g. présentation) en fonction d'un périphérique spécifique par exemple, sans nécessiter la redéfinition de sa structure (e.g. données).

II.3.2.2 Structure d'un document XML

Il existe deux types de documents XML : le document bien formé et le document valide. Un document bien formé est un document respectant la syntaxe XML. Un document valide est un document bien formé qui respecte une structure type définie dans une DTD (sa grammaire).

Un document XML se compose :

- d'un prologue (facultatif mais conseillé),
- d'un arbre d'éléments qui constituent le contenu du document,
- des commentaires et des instructions de traitement.

[37] Par besoin de commodité, XML et SVG seront néanmoins qualifiés de langage dans la suite du document.

Remarque : seul le prologue sera décrit, il permettra de faire le lien avec le dessin proprement dit en langage SVG dans la section II.3.4.

Le prologue précise s'il s'agit d'un document XML. Il identifie le jeu de caractères utilisé et la DTD utilisée. Il peut contenir :

- **une déclaration XML**
  L'exemple ci-dessous donne une déclaration XML :

  > *< ?xml version= "1.0 " encoding = "ISO-8859-1" standalone = "yes" ?>*

  - la partie *<?xml version="1.0"?>* identifie le document XML et la version utilisée,
  - la partie *<?xml version="1.0" encoding="ISO-8859-1"?>* précise le jeu de caractères utilisé (ici l'ISO-Latin 1),

  Remarque : si le jeu de caractères n'est pas précisé, il prend la valeur par défaut (Unicode encodé en UTF-8 ou en UTF-16).
  - la partie standalone='Yes' indique que toutes les recherches de déclarations se feront uniquement dans le document.

- **des instructions de traitement**
  L'exemple ci-dessous donne la syntaxe utilisée pour les instructions de traitement :
  <?nom param1 param2... ?> avec le nom qui identifie une application ou une fonction que l'on appelle en lui passant les paramètres param1, param2, etc.

- **une déclaration de DTD**
  Les exemples ci-dessous donnent deux manières de référencer une DTD :
  <!DOCTYPE racine "url de la DTD"> on précise la location de la DTD via son url.
  <!DOCTYPE racine SYSTEM "racine.dtd"> signifie que la DTD racine se situe dans le même répertoire que le document.

II.3.3 Le langage Java

Le langage Java, qui signifie café en argot américain, a été créé en 1991 par Sun Microsystem dans le but initial de développer des logiciels embarqués pour contrôler des appareils électroniques et leur permettre de communiquer. Ce langage devait permettre de créer des applications sécurisées et surtout exécutables sous diverses plates-formes (Windows, Macintosh et autres) sans modification de l'application grâce à l'utilisation d'une machine virtuelle Java. En 1994, le navigateur Web HotJava permet d'exécuter des programmes Java. Depuis son succès va grandissant grâce à de très nombreuses bibliothèques écrites ainsi que de nombreux environnements de développement (Visual J++, Borland JBuilder, Kawa...) qui lui sont consacrés.

Ce langage semble être bien adapté à l'environnement du Web car les utilisateurs peuvent télécharger du « bytecode » Java depuis Internet et l'exécuter sur des machines multi plates-formes. De plus, le rendu graphique est de plus en plus satisfaisant et plaisant comme nous allons le découvrir dans cette partie très orientée graphisme.

Les classes de base de Java (la JFC pour Java Foundation Classes) sont proposées à partir de cinq grandes bibliothèques :

- AWT pour créer des interfaces utilisateur, peindre des graphiques et des images
- Swing pour améliorer ces interfaces
- Java2D pour créer des images, des dessins complexes (formes, rendu)

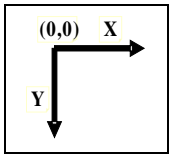
- Drag & Drop pour le transfert de données entre applications Java ou natives
- Accessibility API pour aider les handicapés avec des loupes, des lecteurs de textes...

La bibliothèque Java2D orientée graphisme est donc celle qui sera plus particulièrement développée dans ce chapitre.
De nombreux ouvrages traitent du langage Java (et notamment de Java2D), parmi lesquels [HORST99] [ABPSU] et [HALL01].

II.3.4 Comparaison du dessin2D avec les langages SVG et Java

II.3.4.1 Système de coordonnées

Comme pour toute représentation graphique affichée à l'écran, les technologies SVG et Java utilisent un système de coordonnées pour positionner les éléments. Il comprend un repère qui possède une origine, le coin supérieur gauche du dessin et ses axes (abscisse et ordonnée) comme présentés sur la figure ci-dessous :



L'unité du système de coordonnées sera le pixel en langage Java. SVG pourra l'exprimer en centimètres, en pouces (inches) ou paramétrée par l'utilisateur (e.g. 1 unité = 1 millimètre). Si l'unité n'est pas précisée, elle correspondra au pixel (la valeur par défaut).

Afin d'afficher un résultat à l'écran, SVG utilise les attributs width et height qui donnent respectivement la largeur et la hauteur de la fenêtre. L'attribut viewBox définit la zone visible de ce document. Java différencie aussi l'espace utilisateur de l'espace système.

La figure 29 illustre la représentation d'une fleur de hauteur et largeur fixe (10 et 5) mais en changeant le paramètre de la zone visible (viewbox) :

```
<svg width="10cm" height="5cm" viewBox="0 0 300 200">
<!--code décrivant la fleur--></svg>
```

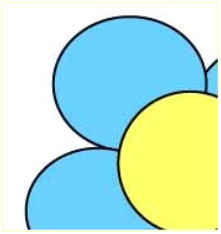
Dessin avec viewBox="0 0 300 200"	Le même dessin avec viewBox="0 0 100 100"
	

Figure 29 : Un même dessin avec une *viewBox* différente.

Cet exemple, même sans entrer dans la programmation permet de voir une des possibilités intéressantes du standard SVG dans une optique de zoom ou de translation, transformations courantes pour les représentations cartographiques.

II.3.4.2 Les formes géométriques de base

Le langage SVG propose un ensemble de balises permettant de réaliser des formes basiques à partir de différents éléments géométriques prédéfinis. Les plus représentatifs sont :

- Le rectangle (balise *rect*), défini à partir des coordonnées du coin haut gauche, par sa hauteur et sa longueur. L'exemple ci-dessous décrit un rectangle positionné à 5 pixels de l'origine (horizontal et vertical) de hauteur 100 pixels et de largeur 50.

  ```
  <rect x="5" y="5" height="100" width="50" />
  ```

- Le cercle (balise *circle*) défini à partir des coordonnées de son centre (cx et cy) et de son rayon r. L'exemple ci-dessous décrit un cercle de centre (5,10) et de rayon 4, le tout exprimé en centimètre.

  ```
  <circle cx="5cm" cy="10cm" r="4cm"/>
  ```

- Le polygone (balise *polygon*), défini à partir d'une liste de coordonnées de points. L'exemple ci-dessous décrit un polygone avec les coordonnées des sommets [(50,100), (150,100),…]

  ```
  <polygon points="50,100 150,100 200,50 120,10>
  ```

- Ainsi que la ligne, l'ellipse (semblable au cercle mais avec 2 rayons pour obtenir un ovale) et la polyligne (ou ligne brisée qui définit une forme ouverte contrairement au polygone).

Tous ces éléments graphiques peuvent recevoir des attributs de style que nous verrons par la suite.
Pour l'instant, nous souhaitons juste dessiner un rectangle jaune avec le langage SVG. Le dessin correspond à la figure 30 ci-après.

Le remplissage de la forme, dans un ton uni, est assuré par la balise *fill.* Il sera appliqué au rectangle précédemment donné en exemple.

SVG possède deux syntaxes différentes pour définir la mise en forme d'un élément :
Le code : *<rect x="5" y="5" height="100" width="150" fill="yellow"/>* a le même effet que le code suivant : *<rect x="5" y="5" height="100" width="150" style="fill:yellow"/>*

On reprend la déclaration XML standard établie dans la section II.3.2.2 (Structure d'un document XML) car SVG est un sous ensemble du langage XML :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

On référence la DTD :

```
<!DOCTYPE svg SYSTEM "svg10.dtd">
```

La racine d'un fichier SVG est un élément <svg>.

```
<?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 30 : Un rectangle jaune uni en langage SVG.

On cherche maintenant à afficher ce même rectangle avec Java :

En langage Java, l'outil de dessin est le *contexte graphique*, objet de la *classe java.awt.Graphics*. Il encapsule l'information nécessaire, sous forme *d'état graphique*.

Deux options sont disponibles pour dessiner cette forme élémentaire :

- L'affichage avec la classe *java.awt.Graphics* bien connue. La figure 31 ci-après (à gauche) montre un objet *g* de la classe *Graphics* auquel on spécifie une couleur à l'aide de la méthode *setColor(Color c)*, ici le jaune (*Color.yellow*). Puis on définit le rectangle tout en le remplissant par la couleur déjà défini grâce à l'instruction *fillRect(5,5,150,100)*.

  Remarque : le rafraîchissement de l'image se fera généralement par la méthode repaint() ou par la méthode update() qui appelle la méthode paint().

- L'affichage avec la *classe java.awt.Graphics2D* (e.g. Java2D) est plus récent et offre plus de possibilités notamment dans le domaine qui nous intéresse, à savoir, la cartographie. La figure 31 ci-après (à droite) décrit le mécanisme d'un affichage avec cette classe :

  La classe *java.awt.Graphics2D* étend la classe *java.awt.Graphics*. Pour utiliser Java2D, il faut transformer l'objet *Graphics* en un objet *Graphics2D*. Par convention, on le nommera *g2* pour ne pas le confondre avec son homologue *g*.

  ```
  public void paintComponent (Graphics g) {
    Graphics2D g2 = (Graphics2D) g;
    ...
  }
  ```

  On choisit l'outil de remplissage (*setPaint(Color)*): ici ce sera une couleur unie en jaune (*Color.yellow*).

  On définit la forme à dessiner Shape dessin : … ; ici un *Rectangle2D*.

  Les classes du package `java.awt.geom` définissent les primitives graphiques tels que des points, des lignes, des courbes, des arcs, des rectangles, et des ellipses.

`Arc2D`	`Ellipse2D`	`QuadCurve2D`
`Secteur`	`GeneralPath`	`Rectangle2D`
`CubicCurve2D`	`Line2D`	`RectangularShape`
`Dimension2D`	`Point2D`	`RoundRectangle2D`

  On décrit la forme avec la commande de type *fill(Shape)*. Le *rectangle2D* sera rempli de jaune uniforme.
  L'affichage proprement dit sera exécuté avec la méthode *paintComponent(Graphics g)*.

Affichage avec Graphics	Affichage avec Graphics2D

public void paint(Graphics g) {
  g.setColor(Color.yellow);
  g.fillRect(5,5,150,100) ;
}
	public void paintComponent(Graphics g) {
	  Graphics2D g2 = (Graphics2D) g;
	  g2.setPaint(Color.yellow);
	  Rectangle2D rect2D =
	  new Rectangle2D.Double(5, 5, 150, 100);
	  g2.fill(rect2D);
	}

Figure 31 : Un même dessin affiché avec les classes *Graphics* et *Graphics2D* de Java.

Le rendu visuel du rectangle Jaune reste identique en utilisant les langages SVG et Java avec les deux méthodes *Graphics* différentes.

Bien entendu, toutes les formes définies préalablement avec le langage SVG sont aussi disponibles sous la forme *Graphics* ou *Graphics2D*.

II.3.4.3 Les formes complexes (ou arbitraires)

En complément des objets graphiques simples, SVG permet de définir des formes complexes. La forme complexe est définie à partir d'un chemin (path en anglais) suivant le concept du point courant. L'attribut de la balise *path* est noté *d* et s'applique en coordonnée absolue (lettre majuscule) ou relative (minuscule) avec les instructions suivantes :

L'attribut *d* caractérise la forme souhaitée grâce à une suite d'instructions :

> *<path d="suite d'instructions" />*

Par analogie avec un stylo, on peut donner des instructions de déplacement (sans écrire) avec la commande *moveto(m)*. Les instructions *lineto(l)* et *curveto(c)* permettent respectivement de dessiner une droite et une courbe toujours à partir du point courant. Et enfin, la forme est éventuellement « clôturer » avec la fonction *closepath (z)*.

L'exemple illustré par la figure 32 décrit une forme complexe. Les couleurs permettent d'identifier la partie de la forme traitée.

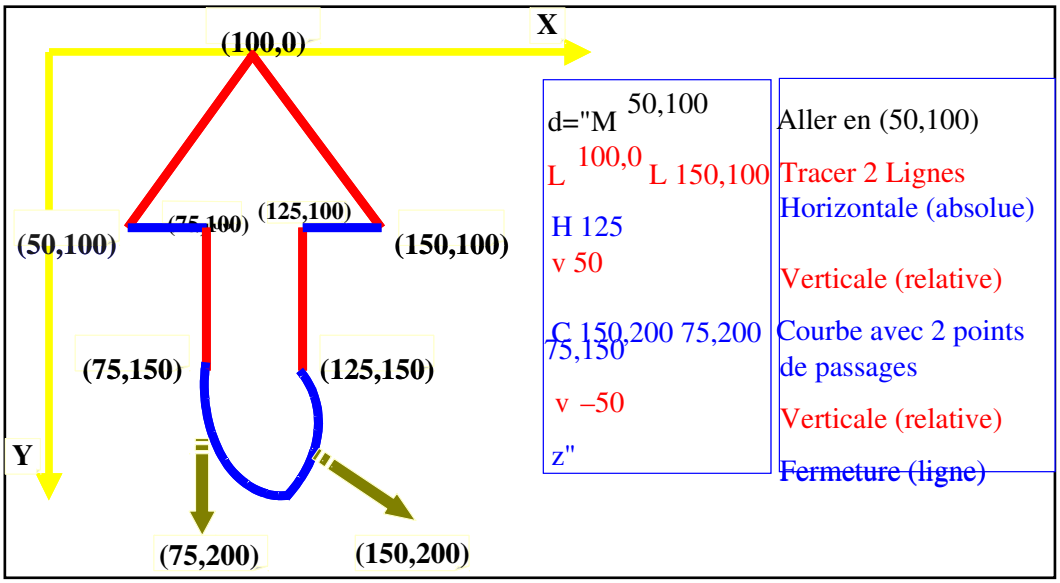


Figure 32 : Une forme complexe en langage SVG.

Avec le langage Java et dans la même optique, une forme complexe (e.g. un polygone) est définie à partir d'un chemin vide :

```
GeneralPath polly = new GeneralPath() ; // création du chemin vide
```

La forme du poylgone est créée à partir de traits reliés entre eux pour définir la forme désirée (e.g. le polygone) grâce à un chemin (GeneralPath polly). Un exemple très simple est présenté ci-dessous :

```
polly.moveTo(5f, of); // premier trait
polly.moveTo(205f, 0f); // second trait
polly.moveTo(5f, 90f); // troisième trait
polly.closePath() ; // on referme par un quatrième trait (plus court chemin)
```

II.3.4.4 Les propriétés des contours

Si nous pouvons remplir une forme avec une couleur, les propriétés de dessin qui s'appliquent au type de trait sont plus intéressantes. En effet dans notre étude, la frontière entre deux unités n'aura pas forcément la même épaisseur, ni la même couleur.

En langage SVG, l'attribut *stroke* définit la forme du bord d'un objet. Ses propriétés sont la couleur (comme vu précédemment pour une forme pleine : *fill :"yellow"*), l'épaisseur du trait (*width*); la forme des angles (arrondi, carré) et la jonction de ligne (*linejoin*).

D'autres propriétés comme la valeur d'opacité (*opacity*) et des propriétés de dégradé seront étudiées plus tard. En effet, le dégradé sur un contour n'apporte pas beaucoup d'intérêt pour notre cause. Tout comme les possibilités de pointillés qui sont délaissés.

L'exemple ci-dessous propose un exemple de jonction de ligne qui peut prendre les valeurs *miter*, *round* ou *bevel*, ainsi qu'une plus grosse épaisseur de trait pour la dernière forme en bleu (*stroke-width*="25px").

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE svg SYSTEM "svg10.dtd" >
<svg width="500" height="500" >
<g stroke-width="10px" stroke="black">
   <rect x="10"  y="50" width="400" height="100" style="fill:yellow"
    stroke-linejoin="miter"  />
   <rect x="10"  y="175" width="400" height="100"  style="fill:magenta"
    stroke-linejoin="bevel"  />
   <rect x="10"  y="300" width="400" height="100" style="fill:cyan"
    stroke-linejoin="round" stroke-width="25px"/>
</g>
</svg>
```

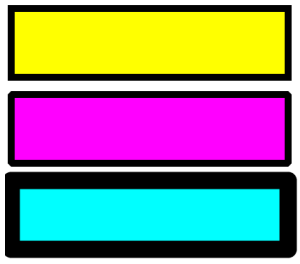


Figure 33 : Propriétés de l'attribut stroke-with et stroke-linejoin en langage SVG.

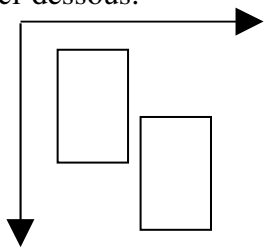
Remarques :

- Afin de factoriser, en vue d'une réutilisation plus aisée du code, l'élément « *g* » permet de regrouper un ensemble d'objets sous une bannière unique. Il est ainsi possible de donner une valeur d'attribut à « *g* » qui sera transmis à l'ensemble de ses éléments constitutifs.
  Dans notre exemple, *stroke-width*="10px" et *stroke*="black" indique l'épaisseur du trait et la couleur souhaitée pour toutes les formes suivantes. Cependant, à même titre que dans la

programmation objet, il y aura héritage de l'attribut, sauf si celui-ci est redéfini. C'est le cas pour la troisième forme avec l'épaisseur du trait.

- Dans un même ordre d'idée, l'élément « *defs* », permet de grouper les objets (comme « *g* ») mais pour les rendre « publiques » par analogie à une fonction afin de les réutiliser par la suite, dans le fichier SVG grâce à l'instruction « *use* » (appel de fonction).
Ainsi, une forme (*rect*)» nommée Rectangle définie une fois peut être réutilisée (*use*) deux fois pour être affichée à deux endroits, comme le montre la figure ci-dessous.

```
<defs>
<rect id="Rectangle" width="60" height="100"/>
</defs>
<use x="10" y="10" xlink:href="#Rectangle"/>
<use x="50" y="50" xlink:href="#Rectangle"/>
```



Avec le langage Java, si en *AWT* (Abstract Windowing Toolkit) toutes les lignes mesurent 1 pixel de large, en Java2D la largeur peut varier en utilisant la méthode *setStroke(Stroke s)*. L'interface *Stroke* est implémentée uniquement par *BasicStroke*. Il est aussi possible, comme SVG, de décrire le rendu d'une forme comme la jonction de deux lignes par exemple.

La classe *BasicStroke(float largeur, int ornement, int jonction)* permet plus précisément de :

- gérer la largeur du trait
- de décrire le type d'ornement souhaité :
  - *BasicStroke.CAP_BUTT* pour une forme carrée ,
  - *BasicStroke.CAP_ROUND* pour une forme arrondie ,
  - *BasicStroke.CAP_ SQUARE* pour une forme carrée « plus longue » .
- de décrire la manière de joindre les extrémités de deux formes :
  - *BasicStroke.JOIN_MITER* joint les segments en étendant leurs bords externes ,
  - *BasicStroke.JOIN_ROUND* arrondi l'angle formé entre deux segments ,
  - *BasicStroke.JOIN_BEVEL* : joint les segments par une ligne droite .

II.3.4.5 Le texte

A partir de données texte, SVG permet de générer des objets graphiques avec tous les paramètres de formatage du texte classique :

- La famille de la fonte
- Le type de fonte
- La taille, la position, la largeur…

Il est aussi possible d'effectuer un certain nombre d'opérations à l'intérieur même de la chaîne de caractères par le biais de la commande « *tspan* » afin d'associer de nouvelles valeurs d'attributs à ce sous ensemble de chaînes comme le montre l'exemple ci-dessous :

```
<text x="10" y="10" style="font-family:Verdana; font-size:12pt; fill:blue">
Bonne chance au prototype <tspan style="fill :red ; font-size:20pt">
```

HYPERCARTE</tspan> !!! - </text>

Bonne chance au prototype HYPERCARTE !!!

Autre effet intéressant qui consiste pour une chaîne de caractères à suivre n'importe quelle forme basique ou complexe. L'exemple complet figure 34 ci-après présente :

- un objet complexe (*d*) défini à partir d'un chemin (*path*) identifié par « chemin »
- les paramètres de notre forme (contour rouge non plein)
- les paramètres de la chaîne de caractères (type de fonte : Verdana, taille : 14.3333 et couleur : bleu)
- la chaîne de caractères « HYPERCARTE … ? » qui va suivre le « chemin » de notre forme complexe (*<textPath xlink:href="#chemin">*)

```
<svg width="5cm" height="6cm" viewBox="0 0 200 175">
<defs>
    <path id="chemin"
    d="M 50,100 L 100,0 L 150,100 H 125 v 50
        C 150,200 75,200 75,150 v -50 z"/>
</defs>
<use xlink:href="#chemin" style="stroke:red;fill:none" />
<text style="font-family:Verdana; font-size:14.3333; fill:blue">
    <textPath xlink:href="#chemin">
    HYPERCARTE - Europe en  SVG . . . Mais Pourquoi pas ?
    </textPath>
</text>
</svg>
```

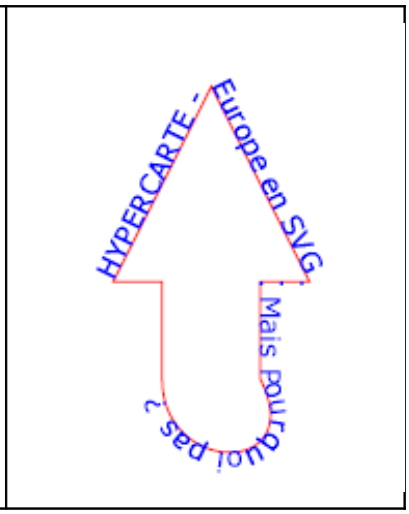


Figure 34 : Un texte qui suit une forme complexe en langage SVG.

Avec le langage Java, le texte sera dessiné à l'écran avec la commande : *drawString(texte, x, y).* Exemple : on définit un objet *f* qui caractérise la fonte, on l'affecte pour l'affichage et enfin on affiche le texte souhaité dans son repère tout en bénéficiant de son contexte d'affichage.

```
Font f = new Font(« TimesRoman », Font.ITALIC, 24) ; // Constructeur
g.setFont(f) ;
g.drawString(« Bonjour « , 30, 30) ; // Bonjour est affiché dans son contexte graphique
                                        (e.g. la fonte, le style et la taille).
```

Il est aussi possible d'établir des effets intéressants avec Java mais ce langage n'offre pas toute la souplesse de SVG dans ce domaine. De plus, en langage SVG, des fonctions de recherche ou de *copier-coller* (vers une autre application) sont possibles car l'affichage (représentation graphique de la donnée) se distingue de la donnée brute, qui demeure en format *Texte*.

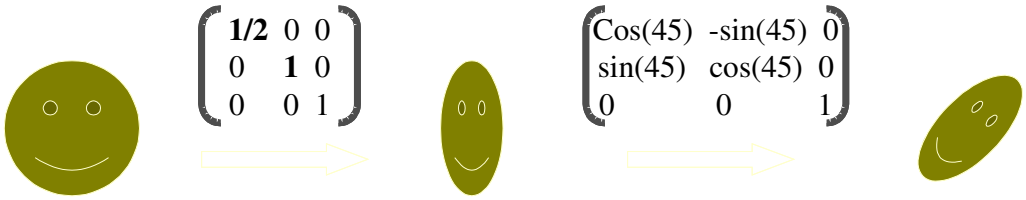
II.3.4.6 Les transformations

Dans le cadre de notre projet, on s'intéresse plus particulièrement aux transformations géométriques que l'on peut représenter grâce à une matrice carrée (3*3) (avec les langages SVG et Java) :

Translation de *tx* sur *x* et *ty* sur *y*	Rotation d 'angle *a*	Dimensionnement de facteur *sx* sur *x* et *sy* sur *y*

$\begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Exemple de transformation pour un redimentionnement de moitié (1/2) sur l'axe des abscisses x puis une rotation de 45 degrés :



Appliquée à une représentation cartographique, ces transformations sont essentielles. En effet, elles sont directement disponibles et apportent d'importantes facilités, ainsi :

- l'utilisateur ne re-calcule pas les nouvelles coordonnées des objets (Translation)
- ni les nouvelles dimensions éventuelles de ces objets (Zoom plus et zoom moins)

SVG met ainsi au service de tous les objets graphiques un attribut « transform » pour définir les transformations à effectuer.

La translation *<g transform="translate(50, 40)">*, permet d'opérer un déplacement du groupe de 50 pixels vers la droite, et 40 pixels vers le bas. Le redimensionnement est donné par la commande *scale(valeur)* et la rotation *par rotate(valeur).*

On retrouve l'équivalent en langage Java grâce aux transformations affines qui servent à modifier les coordonnées utilisateur avant affichage pour, par exemple, centrer le repère au milieu de la zone de dessin.
Les transformations sont la rotation, la translation, la dilatation et le cisaillement (*shear*).

Comme en langage SVG, mathématiquement, une transformation affine est représentée par une matrice 3 x 3 dont la dernière ligne est toujours (0 0 1).
On définit généralement une transformation géométrique entre l'espace utilisateur et l'espace de l'écran avec les instructions :

```
AffineTransfrom at = ...;
g2.transform(at);
```

Plus précisément, la classe *AffineTransform* permet de créer des transformations affines grâce à de nombreuses méthodes :

```
AffineTransform at = new AffineTransform();
at.setToRotation(angle);                  // pour la rotation,
at.setToTranslation(dx, dy);               // pour la translation,
at.setToScale(sx, sy);                    // pour des redimensionnements et
at.setToShear(cx,cy);                     // pour le cisaillement.
```

La classe *AffineTransform* permet aussi de composer des transformations affines :
t.rotate(angle); *t.translate(dx, dy)*; *t.scale(sx, sy)*; *t.shear(cx, cy)*;

Pour les langages SVG et Java, l'ordre des opérations est séquentiel. Chaque transformation aboutit à un nouveau système de coordonnées dans laquelle s'exécutera la deuxième transformation éventuelle et ainsi de suite.
A noter que l'utilisation de *viewBox* pour le langage SVG peut produire les effets similaires d'un zoom ou d'une translation tout comme une applet Java.

II.3.4.7 Effets de rendus

Les deux langages permettent de gérer différents effets intéressants notamment pour la représentation cartographique. Le gradient qui consiste en une transition « douce » entre deux couleurs (linéaire ou radial) et la gestion de la transparence peuvent aisément répondre à une représentation cartographique en contours flous comme le module d'analyse spatiale multiscalaire.

II.3.4.7.1 Dégradés de couleurs

Toute forme ou texte peut se présenter sous la forme de gradient linéaire ou radial. Il existe aussi le remplissage à l'aide d'un motif, mais qui est peu propice à notre problématique.

En langage SVG, l'échelle de couleur pour un dégradé est définie par l'élément *stop* avec la syntaxe suivante :

```
<stop id="name"
    offset="NumberOrPercentage"      qui indique l'arrêt du dégradé
    stop-color="Color"               pour la couleur utlisée à cette limite de dégradé
    stop-opacity="Opacity-value" />  pour l'opacité d'un arrêt de dégradé donné
```

Les valeurs pour l'attribut *offset* sont entre 0% et 100% ou entre 0 et 1.
Pour chaque élément *stop*, la valeur de *offset* doit être supérieure ou égale à la valeur du précédent élément *stop*.

La figure 35 montre le fonctionnement d'un dégradé radial.
Une forme est définie, ici un rectangle ❶, sur laquelle sera effectuée la transformation de rendu. La couleur rouge est utilisée pour remplir le rectangle sous la forme d'un cercle depuis son origine jusqu'a 30% de sa longueur ❷. La couleur jaune est utilisée entre la couronne formée par l'offset à 70% et le restant du rectangle ❸. Entre les deux, de 30 à 70%, la couleur passe graduellement du rouge au jaune. Le dégradé de couleur se situe donc seulement entre ces deux valeurs d'offset.

```
<defs>
    <radialGradient id="Degrade">
      ❷ <stop offset="30%" stop-color="red"/>
      ❸ <stop offset="70%" stop-color="yellow"/>
    </radialGradient>
</defs>
❶ <rect x="300" y="0" width="150" height="150"
style="fill:url(#Degrade)"/>
```

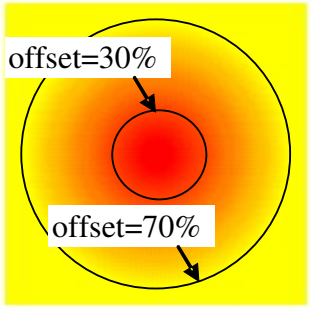


Figure 35 : Fonctionnement d'un dégradé radial en langage SVG.

Le même procédé peut être utilisé avec des gradients linéaires grâce à l'élément *linearGradient* en lieu et place de l'élément *radialGradient*.

Avec le langage Java, un dégradé linéaire est calculé en fonction de deux points de référence dont on fixe la couleur. La couleur des points situés entre les 2 est un dégradé linéaire. Ce dégradé est réalisé grâce au constructeur *GradientPaint* qui se présente ainsi :

> *GradientPaint(x1, y1, color1, x2, y2, color2, boolean);*
> *avec x1, y1, color1 les coordonnées et couleur du point de départ du dégradé*
> *x2, y2, color2 les coordonnées et couleur du point d'arrivée du dégradé*
> *boolean à true pour un dégradé linéaire cyclique (valeur par défaut) sinon un dégradé linéaire acyclique.*

L'exemple (figure 36) montre une ellipse avec une transition de couleur allant du rouge au point de coordonnées (0,0) au jaune de coordonnées (175, 175).
La valeur *true* rencontrée par le constructeur *GradientPaint* signifie que l'opération de rendu va se répéter tout au long de la forme. En son absence, la couleur jaune serait uniforme à partir du point de coordonnées (175, 175) (e.g. dégradé linéaire acyclique).

```
Graphics2D g2 = (Graphics2D)g;
Ellipse2D e = new Ellipse2D.Float(10, 10, 350, 350);
GradientPaint gp = new GradientPaint(0, 0, Color.red,
175, 175, Color. yellow, true);
g2.setPaint(gp);
g2.fill(e);
```

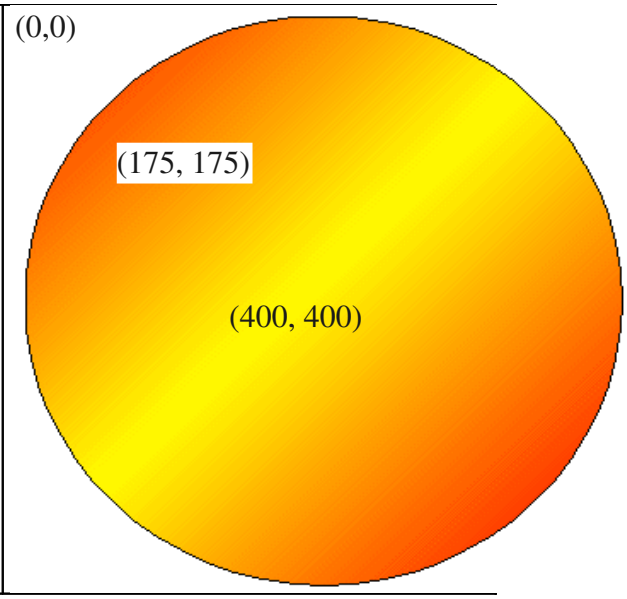


Figure 36 : Fonctionnement d'un dégradé linéaire en langage Java[38].

II.3.4.7.2 Transparence

Le langage SVG permet de gérer différents effets de rendu intéressants notamment pour la représentation géographique. Ainsi, la gestion de la transparence peut répondre à la notion de couches plus ou moins apparentes. Le calcul des différentes couches peut intervenir une seule fois, tandis que l'utilisateur cherchera à visualiser une ou plusieurs couches superposées en ne jouant que sur le paramètre d'opacité.

L'exemple figure 37 montre un rectangle rouge avec son contour noir ❶ à l'horizontale. Les cinq autres rectangles en position verticale sont bleus avec un contour cyan ❷. Le paramètre de transparence (e.g. opacity) va s'appliquer sur ces cinq rectangles en variant de 0.2 pour le rectangle le plus à gauche ❸ jusqu'à la valeur 0.8 pour l'avant dernier rectangle à droite ❹. Le

[38] L'image provient du site http://www.apl.jhu.edu/~hall/java/

dernier rectangle à droite ❺ n'ayant pas de paramètre d'opacité, il sera considéré (par défaut) comme 1.0 pour totalement opaque.

```
❶ <g style="stroke:black;fill:red" >
     <rect x="10"  y="150" width="400" height="100"/>
  ❷ <g style="stroke:blue;fill:cyan">
     ❸ <rect x="25"  y="50"  width="50"  height="320"
        style="stroke-opacity:0.2;fill-opacity:0.2"  />
        <rect x="100" y="50"  width="50"  height="320"
        style="stroke-opacity:0.4;fill-opacity:0.4"  />
        <rect x="175" y="50"  width="50"  height="320"
         style="stroke-opacity:0.6;fill-opacity:0.6"  />
     ❹ <rect x="250" y="50"  width="50"  height="320"
         style="stroke-opacity:0.8;fill-opacity:0.8"  />
     ❺ <rect x="325" y="50"  width="50"  height="320" />
    </g>
  </g>
```

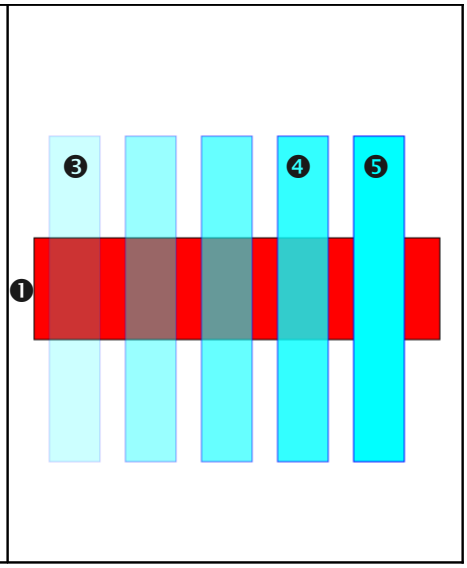
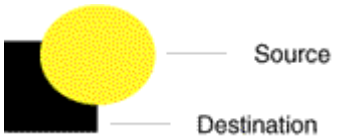


Figure 37 : Gestion de la transparence en langage SVG.

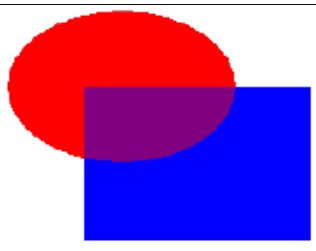
En langage Java, la gestion de la transparence est rendue possible grâce à la notion de « compositing » qui permet de mélanger les couleurs de plusieurs formes selon le choix d'une règle et de gérer la transparence en spécifiant une valeur alpha.

La règle permet de choisir parmi tous les modes de composition de l'image construite avec l'image existante, représentée par des constantes de la classe *AlphaComposite*. Quelques compositions sont montrées en exemple[39] ci-après.

Source-over (SRC_OVER)	Destination-in (DST_IN)	Clear (CLEAR)
		

Le coefficient alpha spécifie le degré de transparence désiré de la valeur 0.0 pour totalement transparent à 1.0 pour une complète opacité. Si « source » représente l'alpha de la source et « destination », l'alpha de destination alors la valeur apha de la zone commune aux deux formes est alors source (1-destination) ou destination (1-source).

La valeur d'alpha ne change pas les modes de composition, il atténue l'impact de l'image construite comme le montre la figure 38.

Une ellipse rouge est dessinée en alpha 1 (opaque) et, selon le choix de la règle de composition (e.g. rule), la compose avec un rectangle bleu. Le style de composition appliqué au rectangle est choisi par : Composite composite = AlphaComposite.getInstance(rule, alpha); g2.setComposite(composite);	

[39] Les images proviennent du site http://java.sun.com/docs/books/tutorial/2d/

avec rule = AlphaComposite.SRC_OVER et alpha = 0.5 pour 50% de transparence.	

Figure 38 : Gestion de la transparence en langage Java.

Si la composition des formes ci-dessus se base sur un « mélange » de couleurs. Il est possible de combiner les formes elles mêmes. La « Constructive area geometry » (CAG) est le procédé pour créer de nouvelles formes complexes en utilisant une logique booléenne. Les exemples figure 39 montre les possibilités d'opérations sur des types *Area* :

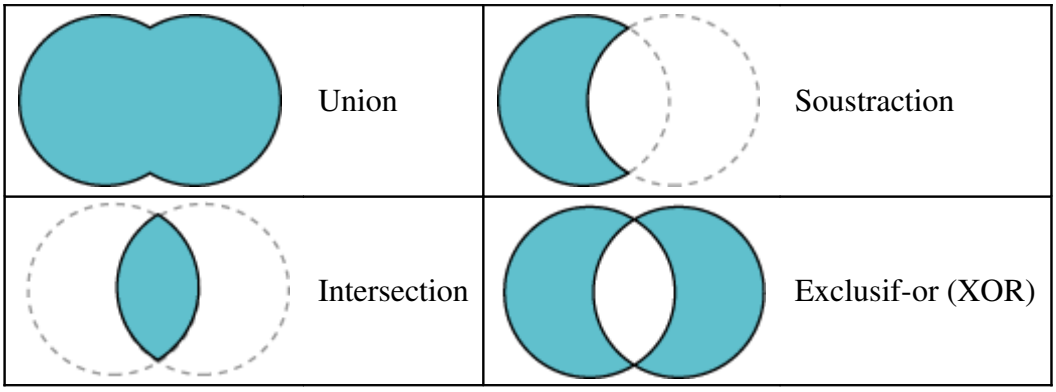
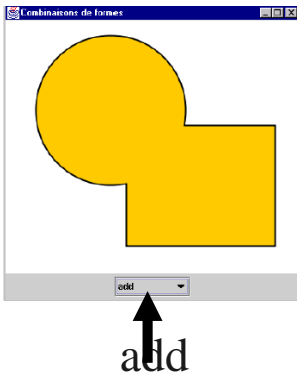


Figure 39 : La « Constructive Area Geometry » en langage Java.

Il est ainsi aisé d'imaginer des possibilités logicielles, par exemple pour fusionner des zones géographiques constituant un même territoire, etc.

L'exemple 40 montre un exemple de codage possible utilisant le procédé de la *Constructive area geometry*.

```
public void paintComponent(Graphics g) {
  Graphics2D g2 = (Graphics2D)g;
   ...
   Area areaOne = new Area(ellipse);
   Area areaTwo = new Area(rectangle);
   if (option.equals("add")) areaOne.add(areaTwo); // Union
   else if (option.equals("intersection"))    areaOne.intersect(areaTwo);
   else if (option.equals("subtract")) areaOne.subtract(areaTwo);
   else if (option.equals("exclusive or")) areaOne.exclusiveOr(areaTwo);
   g2.setPaint(Color.orange);
   g2.fill(areaOne);
   g2.setPaint(Color.black);
  g2.draw(areaOne);
}
```



add
(correspond à l'union des 2 formes)

Figure 40 : Utilisation de la CAG en langage Java.

II.3.4.8 Les interactions

Les interactions avec java2D ne peuvent faire l'objet d'une étude exhaustive, le sujet étant trop vaste. Il en est de même avec le langage SVG [SVG]. Cependant, un état des lieux sommaire peut être établi.

Le langage SVG permet d'obtenir des cartes interactives par le biais de gestion d'évènements, ces fonctions interactives peuvent être liées :

- aux périphériques d'entrée tel que la souris et le clavier quand l'utilisateur :
    - sélectionne du texte
    - sélectionne un objet, le désélectionne
    - positionne sa souris sur un objet
- aux traitements de plus haut niveau du langage :
    - le fichier SVG est ouvert par une application ou fermé
    - une application redimensionne l'image, l'a fait déplacer

Une particularité remarquable du langage SVG permet, après installation d'un plug-in de viewer SVG[40] , d'obtenir directement certaines fonctions interactives déjà intégrées comme :

- le zoom avant/arrière
- le retour à la vue initiale (panoramique)
- le choix de qualité supérieure ou non (lissage)
- d'afficher le document source SVG
- de copier le code source du SVG (dans le presse-papier)
- d'enregistrer le code source du SVG (fichier original)

A cette panoplie d'interactions existantes, il faut bien sûr intégrer toutes les possibilités via l'utilisation des langages Javascript ou Java. Car SVG possède une interface avec DOM (Document Object Model) permettant la représentation d'une arborescence XML en mémoire sous forme de nœuds. Chaque nœud est un objet muni de méthodes comme lire, modifier, supprimer et coller. Le développeur accède donc à tous les éléments du document SVG (et donc ses attributs) permettant un contexte dynamique et interactif utile pour notre projet.

II.3.5 Conclusion

Les objets graphiques obtenus grâce au métalangage SVG sont de grandes qualités visuelles. Le traitement de type vectoriel n'y est pas étranger. Les objets définis peuvent être regroupés, transformés, composés dans d'autres objets et bien sûr recevoir des attributs de style.
Les graphiques SVG sont donc interactifs et dynamiques. Cette approche peut aisément rendre de nombreux services dans le cadre de la représentation cartographique. L'animation peut être définie soit à l'intérieur des fichiers SVG, soit dans un langage de script externe. N'oublions pas que faisant partie de la grande famille XML, SVG possède une interface (avec DOM notamment) afin d'accéder à tous les attributs des éléments de la hiérarchie ouvrant encore de nombreuses possibilités liées à une programmation objet classique. Cependant, tout ce que peut réaliser le langage SVG, Java2D peut le faire via un peu de programmation. Les deux alternatives comme nous avons pu le constater offrent des fonctionnalités similaires de haut niveau. On ne peut donc raisonnablement pas les opposer, mais peut être chercher à les faire cohabiter.

[40] Exemple le Viewer d'Adobe

III RÉALISATION

La réalisation du prototype ATM, composante du projet Hypercarte, impose les différentes phases que l'on retrouve dans le domaine du génie logiciel. Ces phases visent à concourir à un développement optimal en respectant les contraintes du projet.
Tout d'abord, on cherchera à définir une architecture capable de répondre à notre problématique d'échanges d'informations à distance. Celle-ci sera précisée lors des choix technologiques. Elle sera enrichie de choix d'ordre plus conceptuel permettant d'introduire la phase de modélisation du module. Enfin, la réalisation proprement dite sera effectuée conformément aux choix de conceptions préalablement établis. Cette phase d'implémentation sera observée à partir de copies d'écrans du prototype complétées d'extraits de codes significatifs.

III.1 Choix technologiques et conceptuels

Le module ATM doit permettre de visualiser des cartes en fonction de critères définis par les utilisateurs via le réseau Internet. Ce type d'application influence naturellement des choix technologiques et conceptuels au sein d'une architecture capable de répondre à nos attentes.
Ces attentes concernent, en premier lieu, les échanges entre une interface utilisateur côté client et un serveur distant. Les réponses apportées devront introduire le type d'architecture générale de l'application à utiliser (client léger ou lourd, donnée cartographique à diffuser, etc.) conditionnant ainsi des choix technologiques, tandis que les réflexions liées à la structuration des données et à leur accès sur le serveur (interrogation) font plus particulièrement référence à des choix de conception.

III.1.1 Choix technologiques

III.1.1.1 Quel type de client ?

Nous avons observé dans le domaine des solutions appliquées à la cartographie interactive sur Internet (chapitre II.2) que l'utilisation de plusieurs machines distantes (client et serveur) impose une réflexion sur la répartition des tâches entre ces entités. Cette réflexion s'appuie fondamentalement sur deux types de stratégies qui conditionnent l'architecture à déployer.
Dans notre application, une architecture client léger impliquerait une très grande puissance de calcul de la part du serveur pour obtenir généralement une carte faiblement interactive (e.g. carte au format raster). Si l'on tient compte de notre volonté de distribuer massivement l'application, cette puissance ne sera pas forcément disponible. D'autre part, une simple analyse même rudimentaire entraînerait de grands transferts de données entre le client et le serveur pour généralement re-calculer toute la carte. C'est pourquoi on décide naturellement de transférer de « l'intelligence » au navigateur pour obtenir un **client lourd plus autonome**.

III.1.1.2 Quelle technologie pour le client ?

Le choix entre la technologie ActiveX ou Java (section II.2.3.3) repose en partie sur le niveau de compatibilité recherchée. L'**applet Java** est préférable pour offrir une diffusion à grande échelle grâce à ses caractéristiques d'exécution multi plates-formes.

III.1.1.3 Répartitions des tâches client-serveur

Si l'utilisation d'une applet Java indique que notre client lourd sera susceptible d'opérer des opérations de traitement de manière autonome, il est nécessaire de préciser lesquels.
En effet, une première réflexion peut conduire à penser qu'à la première connexion avec le serveur, celui-ci va envoyer au client le programme et toutes les données brutes nécessaires aux

traitements. Cette stratégie s'apparente à une solution Java indépendante (section II.2.3.3) car toutes les opérations (affichage, analyse, etc.) sont gérées par le client. Cependant, cette optique est contestable car elle impose aux clients un chargement complet des ressources disponibles (non forcément nécessaire). D'autre part, si afficher les informations d'une unité cliquée à la souris ne nécessite pas une puissance de calcul démesurée, il en est tout autre pour les traitements d'analyse spatiale qui font intervenir un grand nombre de relations (section II.1.5.2). Ainsi, pour ne pas provoquer une forme de rejet d'une application qualifiée de « lente », la solution retenue est de **pré-calculer**, par le **serveur**, l'ensemble des **données nécessaires pour chaque unité**. L'« objet » cartographique transmis, le client utilisera ses services (e.g. méthodes) pour connaître le nom de l'unité, avec quelle unité elle est contiguë, etc. Dès lors, le client sera capable de procéder localement aux différentes analyses et fonctionnalités demandées à la cartographie interactive (zoom, déplacements, analyses etc.) dans des temps raisonnables.

III.1.1.4 Echanges et affichage des données cartographiques

Les échanges entre client et serveur impliquent le choix du type de format de données à transférer. Une image au format raster est pauvre en qualité graphique et subit des dégradations lors d'un zoom par exemple, alors que le format vecteur offre des qualités graphiques irréprochables. Il bénéficie en outre d'une grande richesse conceptuelle qui sera exploitée par le langage (section II.2.3.2). On choisira donc une **image au format vectoriel**.

Cependant entre les solutions Java avec sa classe Graphics2D et le standard SVG, le choix est plus difficile. Ils bénéficient tous deux de qualités comparables en rendu comme en interactions (section II.3.5).
La solution de transférer un fichier SVG apporte un avantage sur la structure arborescente de l'objet qui est déjà réalisée (e.g. hiérarchie XML). Cependant, la taille du fichier, même s'il est compressible est relativement importante. Il impose aussi un long parcours des éléments de la hiérarchie pour être chargé en mémoire et exploité chez le client par l'applet Java. La solution retenue est donc de **transmettre les objets en format natif Java sérialisés** beaucoup plus légers (e.g. code binaire), d'où une meilleure utilisation de la bande passante. La dé-sérialisation qui consiste à charger les objets en mémoire vive chez le client est très rapide et ne nécessite aucun traitement particulier. L'**affichage** de la **donnée cartographique** sera alors réalisée grâce à la **classe Graphics2D** de Java.

III.1.1.5 Choix d'un serveur et d'une base de données

Le serveur pour diffuser l'application et les données est disponible au sein de l'équipe de recherche de l'INRIA (Projet APACHE [APACH]). L'architecture particulière du serveur (parallélisme), présenté en figure 41, sert d'expérimentation à la programmation et aux outils d'exploitation des machines parallèles.

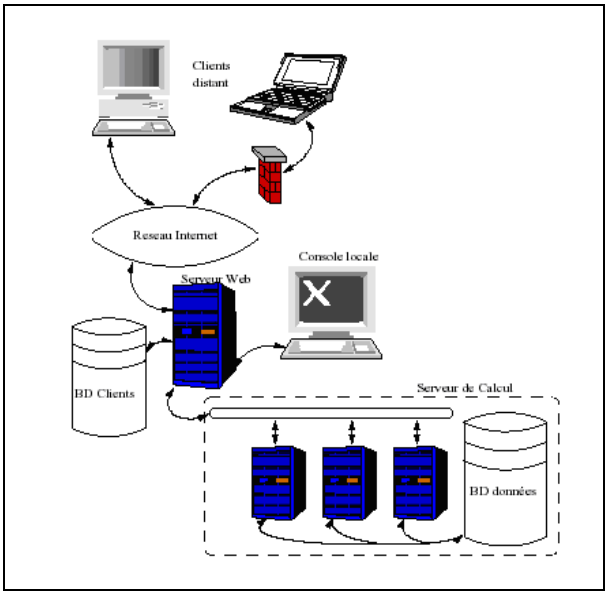


Figure 41 : Architecture générale du serveur.

L'architecture cible est un serveur dont le module de calcul est une grappe de PC. Le frontal est la première structure du serveur qui reçoit les requêtes. Il contient un cache qui lui permet de garder les traces des requêtes déjà exécutées et de conserver des résultats intermédiaires acquis. Le frontal transmet les requêtes non présentes dans son cache au serveur de calcul après les avoir transformées en graphe de tâches pour un traitement en parallèle. L'utilisation du langage de programmation parallèle Athapascan [DOREI99] permet de construire automatiquement le graphe de tâches en utilisant notamment les relations de précédence imposées par le programme. Il ordonne dynamiquement ce graphe sur la machine cible, dans notre cas une grappe de PC (de l'ordre de 60 nœuds).

Cette architecture a déjà permis la réalisation de cartes expérimentales produites à partir du module d'analyse spatiale multiscalaire du projet Hypercarte. Elle sera à terme utilisée pour notre module ATM afin d'optimiser les calculs à travers la mise en place de serveurs cartographiques interactifs sur serveur Web.

Cependant, les fichiers nées fournis par l'équipe du laboratoire Géographie-cités n'ont pas été intégrés dans une base de données à ce stade expérimental (e.g. structure de données non stabilisée). C'est pourquoi on utilisera dans un premier temps (e.g. prototype) le serveur comme une simple entité de stockage capable de délivrer l'applet et les données Java. Il sera néanmoins capable d'intégrer de nouvelles données ou des mises à jour par traitement autonome afin de les rendre disponibles pour notre applet.

Cette « contrainte » liée au stockage de l'information apporte néanmoins deux modes de déploiement possibles de notre solution. La première avec un serveur Web classique engendre un potentiel de diffusion important lié à un déploiement facilité, tandis que l'utilisation d'un serveur cartographique (plus performant) reste plus spécifique et s'adresse donc à une clientèle moins nombreuse.

III.1.1.6 Environnement de développement

Le kit de développement Java par défaut est le JDK pour Java Development Kit. Il comprend plusieurs outils d'aide au développement d'applications Java et existe en plusieurs versions compatibles pour un grand nombre de plates-formes (Unix, Solaris, Windows…). Il est proposé en téléchargement gratuit par Sun, ce qui a permis une diffusion très rapide du langage. Même si le JDK comprend de non outils puissants, le développement d'applications importantes impose un passage à des environnements plus évolués.

JBuilder de Borland [BORLA] propose l'ensemble des fonctions classiques d'un outil de développement traditionnel : depuis un module RAD (Développement Rapide d'Applications), en passant par un navigateur Web intégré, et la compilation de code. De plus, il offre les principales caractéristiques d'un IDE (Environnement de Développement Intégré), soit un éditeur de code source et un gestionnaire de projets. Ces nombreuses fonctionnalités ont contribué à la très bonne réputation dont il jouit au sein de la communauté des développeurs [DEVEL].

Le produit **JBuilder** 8 en version personnelle est choisi pour **implémenter le projet**. Il nous facilitera notamment l'utilisation de la **bibliothèque Swing** pour la réalisation de notre **interface graphique** permettant des requêtes utilisateurs visuelles, tandis que l'API Java2D qui fait aussi partie de la bibliothèque Java Foundation Classes nous permettra l'affichage des résultats sous forme de cartes.

III.1.2 Choix conceptuels

Nous avons choisi de rendre autonome l'objet cartographique chez le client (section III.1.1.3). Le serveur doit donc traiter les objets (e.g. unités territoriales) et fournir un ensemble de méthodes pour y accéder depuis l'interface. Cette partie concerne plus particulièrement le travail de l'équipe du laboratoire ID-IMAG avec laquelle mon équipe a collaboré. Sans être exhaustif, on décrit la démarche générale du travail réalisé à partir des fichiers sources décrits au paragraphe I.4. (section : Fichiers de données mis à disposition) qui influent en partie sur les solutions choisies pour cette étude (section I.4 section : Problématique soulevée).
Cette démarche, présentée par typologie de carte, introduit la définition de méthodes données à titre indicatif pour préciser la réponse à nos différents objectifs. Ces méthodes seront précisément identifiées dans le cadre des échanges avec l'interface dans le chapitre suivant.

III.1.2.1 Phénomènes d'emboîtement hiérarchique : Analyse hiérarchique

L'analyse multiscalaire implique la comparaison entre (au moins) deux niveaux d'observation différents. On veut ainsi comparer une région (e.g. un niveau) par rapport à son pays d'appartenance (e.g. un autre niveau) sur une variable statistique donnée.

L'ensemble des unités territoriales devra intégrer la connaissance de cette structure hiérarchique. Elle est décrite par le fichier *UT_SUP.txt* qui décrit la hiérarchie d'emboîtement d'une UT et de son unité hiérarchique supérieure. La figure 42 montre un aperçu de cette structure appliquée aux unités territoriales.

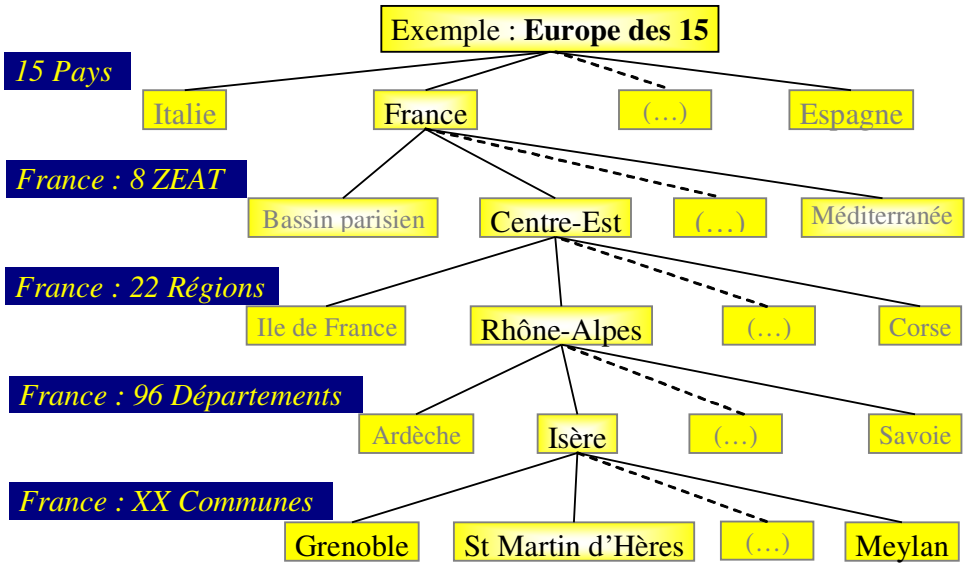


Figure 42 : Structure hiérarchique d'emboîtement

La description de cette hiérarchie d'emboîtement sera définie grâce à :

- La méthode « UT := ton_Unite_Supérieure() », appliquée à l'UT en question retourne de 0 à 1 UT. On choisit donc de parcourir l'arbre de bas en haut.

L'appartenance de l'objet cartographique à un espace donné (e.g. Europe des 15, Europe élargie, etc.) ainsi qu'à un découpage (un pays, une région, etc.) est décrite par le fichier *ESPACE.TXT* .

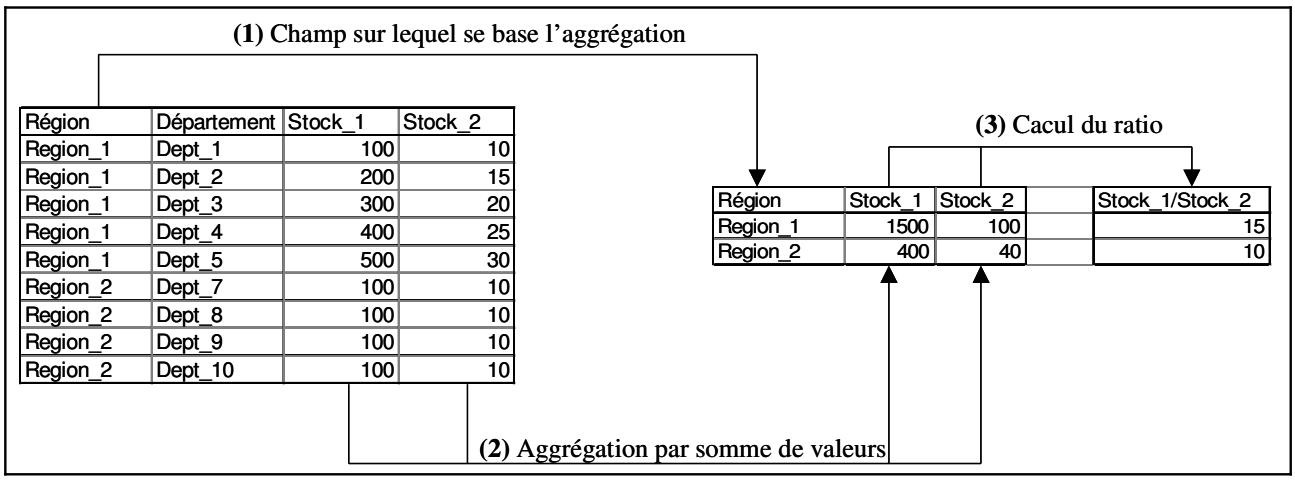
- La méthode « Boolean := est_Present(Espace, Découpage) » avec comme paramètres le nom de l'espace et le nom du découpage retourne vrai si l'UT appartient à ces deux entités.

Les données statistiques (e.g. thématique) sont fournies, au seul niveau élémentaire, par les fichiers Pib99.txt et Population99.txt accessibles par :

- La méthode « Int := ton_Stock(Chaîne) » avec en paramètre le nom du stock souhaité.

Le lieu de collecte des informations des données statistiques étant dans notre étude l'unité élémentaire, ce sera donc nécessairement les stocks de ce niveau (N) qui seront agrégés dans un premier temps, puis cumulés au niveau supérieur (N-1) jusqu'au niveau le plus haut référencé dans notre hiérarchie comme l'exemple décrit dans la section II.1.5.2.2.3.
Nous cherchons aussi à effectuer des calculs de ratios. Cette opération d'agrégation thématique n'est pas anodine dans le cas de données quantitatives. En effet, pour produire de nouveaux indicateurs (e.g. ratio de 2 stocks) [DAO03], il est nécessaire de toujours agréger les informations élémentaires (cumul des stocks à partir des unités au niveau spécifié) et enfin de réaliser le ratio demandé. Un champ de type ratio pré calculé dans une modélisation n'aurait donc pas de sens pour l'agrégation.

La figure ci-dessous montre, chronologiquement, les opérations à effectuer pour ne pas effectuer de calcul erroné.



Région	Département	Stock_1	Stock_2
Region_1	Dept_1	100	10
Region_1	Dept_2	200	15
Region_1	Dept_3	300	20
Region_1	Dept_4	400	25
Region_1	Dept_5	500	30
Region_2	Dept_7	100	10
Region_2	Dept_8	100	10
Region_2	Dept_9	100	10
Region_2	Dept_10	100	10

Région	Stock_1	Stock_2	Stock_1/Stock_2
Region_1	1500	100	15
Region_2	400	40	10

III.1.2.2 Phénomènes de voisinage : Analyse spatiale

L'analyse multiscalaire de notre étude implique aussi la notion de contiguïté entre unités voisines. Les types de traitements engendrés font partie de l'analyse spatiale et intègrent nécessairement la notion de topologie. La topologie permet d'indiquer que deux unités sont contiguës si elles partagent une frontière commune. La figure 43 indique que la région I1 est directement voisine (e.g. contiguës) avec les régions I3 et I4.

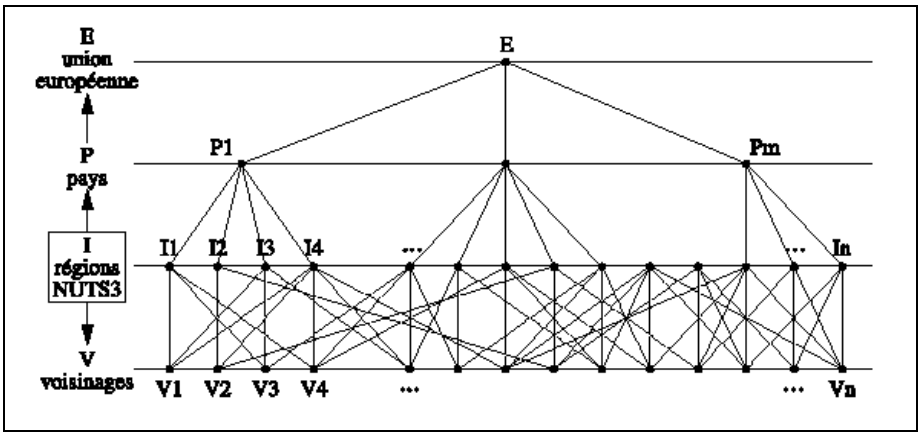


Figure 43 : Voisinage des régions en Europe [MATPI02].

Si la notion de contiguïté est généralement rendue explicite au sein d'une base de données (section II.1.5.2.2), aucun fichier disponible ne fournit d'information sur la topologie, elle doit donc nécessairement être calculée. Les fondements de l'approche choisie sont développés ci-dessous.

- Le fichier de géométrie *map.txt* est donné sous la forme *Code_UT*, *X*, *Y*, et *Num_Polygone*,

avec *Code_UT* l'identifiant de l'unité, *X* et *Y* les coordonnées de chaque point composant le polygone et *Num_Polygone* correspond au numéro de polygone de l'UT.
Un extrait de ce fichier est donné ci-dessous :

Code_UT	X	Y	Num_Polygone
RA	10	5	1
RA	**10**	**15**	1
RA	20	15	1
RA	30	5	1

...	…	…	
RB	5	10	10
RB	**10**	**15**	10
RB	20	20	10
...	…	…	...

A partir du fichier *map*, on constitue deux fichiers (*points* et *geometrie*) :

- Le fichier *points.txt* qui identifie chaque point (en coordonnées *X* et *Y*). Il est composé de trois champs : *Id_Point, X, Y.*

Id_Point	X	Y
1	10	5
2	10	15
3	20	15
4	30	5
…	…	…
100	5	10
101	20	20
...	…	…

Principe du traitement : le fichier *map* est parcouru de haut en bas. Pour chaque enregistrement, on vérifie si le couple (X, Y) est déjà identifié dans le fichier *point*. Si ce n'est pas le cas, on créé un nouvel identifiant dans le fichier *point* et on affecte les valeurs *X* et *Y*.

- Le fichier *geometrie.txt* qui donne pour chaque UT du fichier *map* l'ensemble des identifiants des points la constituant. Il est donc composé d'autant de champs qu'il est nécessaire : *Code_UT, Id_Point, Id_Point, etc.*

Code_UT	Id_Point	Id_Point	Id_Point	Id_Point
RA	1	**2**	3	4
RB	100	**2**	101	
…	…	…	…	…

***Principe du traitement* :** le fichier *map* est à nouveau parcouru de haut en bas. Pour chaque enregistrement, on recherche l'identifiant correspondant au couple de valeur (*X*, *Y*) dans le fichier *point*. Puis on l'enregistre dans le fichier *geometrie*.

L'étape finale pour l'intégration de la topologie consiste simplement à parcourir le fichier *geometrie* et à affecter dans un nouveau champ le ou les identifiants des UT voisines de l'UT en cours.

Code_UT	Contigue	Id_Point	Id_Point	Id_Point	Id_Point
RA	RB	1	**2**	3	4
RB	RA	100	**2**	101	
…		…	…	…	…

Les unités *RA* et *RB* partagent le même identifiant de points (e.g. 2), donc les mêmes coordonnées du point. Elles sont donc considérées comme contiguës.

La procédure est (très) coûteuse en temps, mais elle est réalisée une seule fois sur le serveur. D'autre part comme elle est pré-calculée, elle est nécessairement plus performante qu'une structuration classique dans un SGBD (e.g. nombreux liens).

La notion de topologie de type contiguïté est fournie grâce à :

- La méthode « {UT} := Tes_Unites_Contigues() », appliquée à l'UT en question retourne de 0 à n unités contiguës à l'unité traitée.

Les données de géométrie qui serviront à tracer la forme de l'unité pour l'affichage seront accessibles par :

- La méthode « Area := ta_surface() », appliquée à l'objet UT qui retourne une surface constituée à partir de points (e.g. fichier *geometrie*).

Remarque : lors de l'étape de création du fichier *points*, il est possible d'intégrer directement l'identifiant Code_UT dans la table comme ci-dessous. La contiguïté est ainsi directement identifiée mais seulement au niveau du point. Ce qui n'est pas satisfaisant.

Id_Point	X	Y	Code_UT
1	10	5	RA
2	10	15	RA, RB
3	20	15	RA
4	30	5	RA
…	…	…	…
100	5	10	RB
101	20	20	RB
...	…	…	

Le traitement du fichier ci-dessus serait cependant plus performant que l'étape de création du fichier *geometrie* précédente (en partant de Code_UT). Mais dans ce cas, l'ordre des points identifiés ne serait plus respecté (voir ci-dessous).

Situation précédente (ordre respecté pour RB)					Situation actuelle (ordre non respecté)				
Code_UT	Id_Point	Id_Point	Id_Point	Id_Point	Code_UT	Id_Point	Id_Point	Id_Point	Id_Point
RA	1	2	3	4	RA	1	2	3	4
RB	100	**2**	101		RB	**2**	100	101	
…	…	…	…	…	…	…	…	…	…

L'ordre des points qui constituent une forme complexe n'est pas commutatif pour l'affichage. On gardera bien sûr notre première version du fichier *geometrie*.

III.1.2.3 Synthèse

Les unités territoriales sont « autonomes » chez le client, elles intègrent donc toutes les valeurs statistiques disponibles, le contour polygonal pour afficher la carte, la liste des unités contiguës pour les notions de voisinage, l'unité territoriale de niveau immédiatement supérieur (hiérarchie d'emboîtement) et les notions d'appartenance à un espace et à un découpage donné.
Le schéma ci-dessous (figure 44) décrit la modélisation minimale retenue pour la partie donnée du prototype.

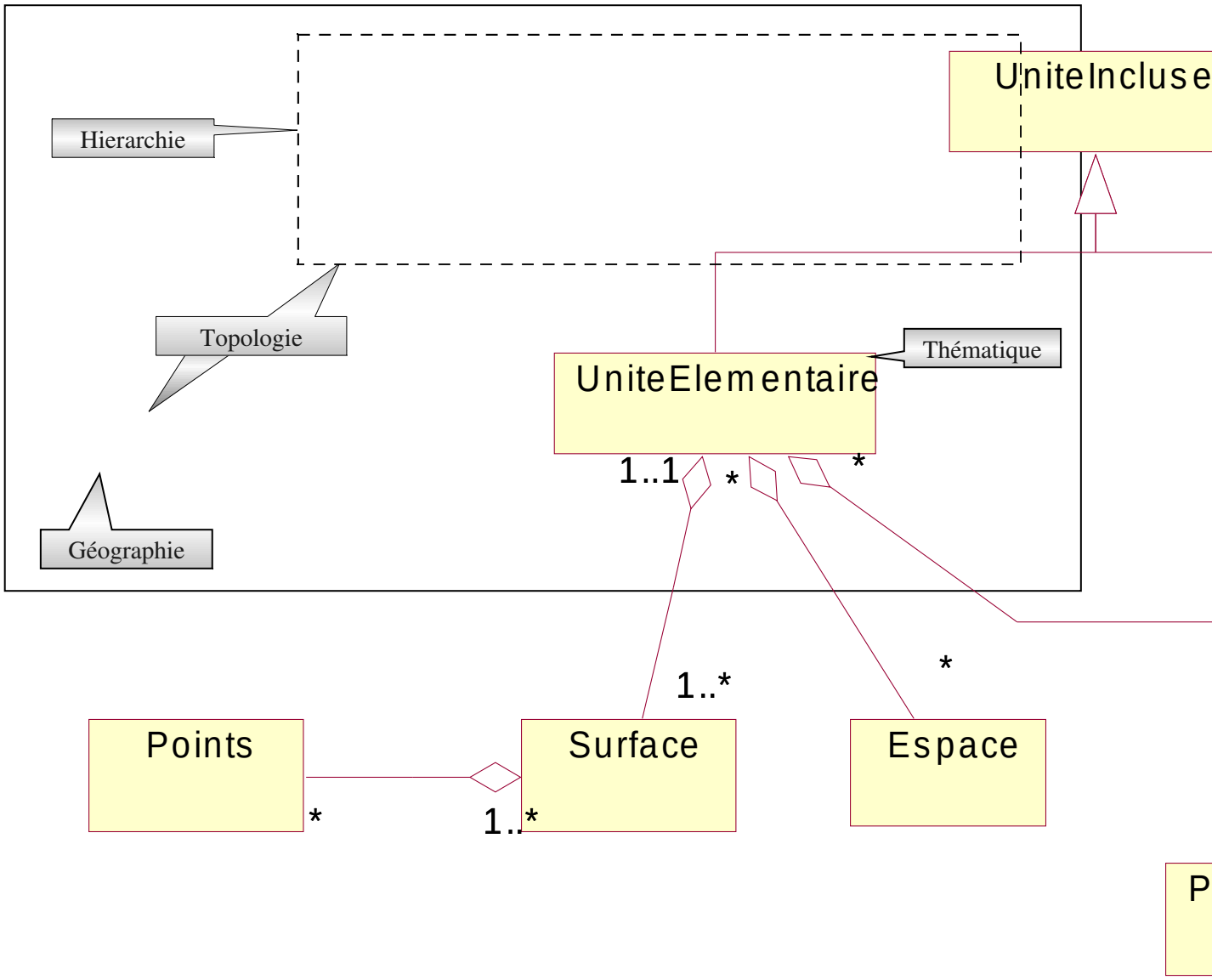


Figure 44 : Modélisation simplifiée du package Hypercarte.Data

Les unités territoriales non élémentaires sont composées :

- d'autres UT composant une **hiérarchie** d'emboîtement.

Cette hiérarchie correspond à une application du composite de Gamma. Le formalisme dit de Gamma comporte 14 rubriques permettant de décrire les patrons de conception[41] proposés dans le catalogue [GAMMA95].

Les unités territoriales élémentaires (source de l'information) sont composées :

- d'informations **thématiques** (population, PIB),
- d'informations d'appartenance à des espaces,
- d'informations **géographiques** (des surfaces composées de points)

La **topologie** (e.g. analyse spatiale) prend en compte au moins deux unités territoriales. Elle est déduite sur les seules données de géométries (coordonnées spatiales des points donnés en *X* et *Y*).

Ces données seront accessibles via une API qui fournit un mode d'accès à une bibliothèque de fonctions pour obtenir les informations désirées sur les unités. Le package « Hypercarte.Data » écrit par Said Oulahal de l'équipe du laboratoire ID-IMAG devrait permettre ces accès.

III.1.3 Conclusion

L'architecture fonctionnelle du processus d'exploitation de notre prototype est schématisée dans la figure 45.

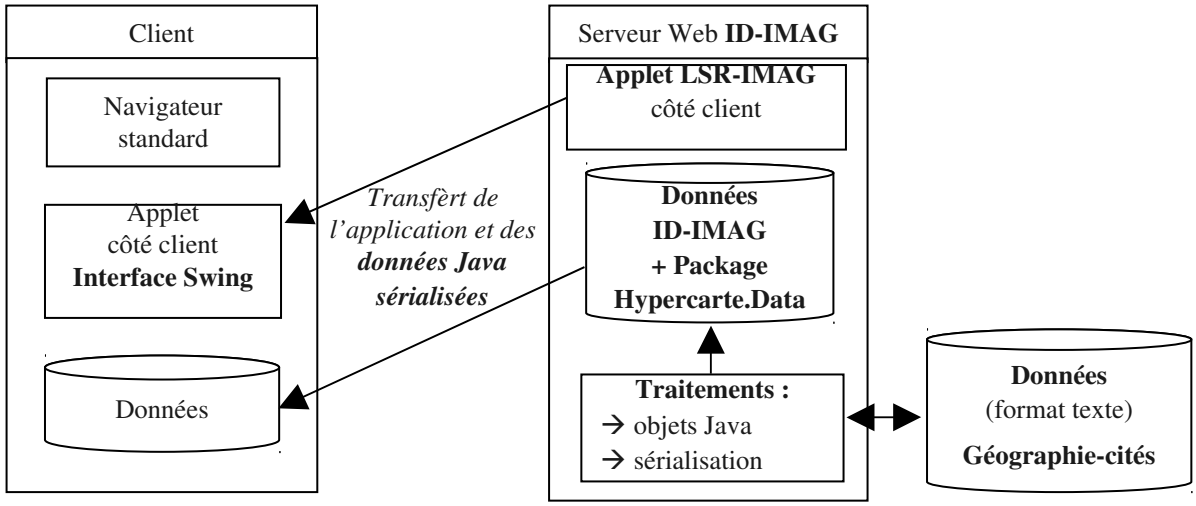


Figure 45 : Architecture fonctionnelle retenue pour le prototype ATM

Les données (en format texte) de l'équipe du laboratoire Géographie-cités sont fournies à l'équipe du laboratoire ID-IMAG. Celle-ci les traite pour intégrer automatiquement les notions de contiguïté, de hiérarchie etc. Les différentes données, notamment cartographiques au format vectoriel (Java2D), seront accessibles ultérieurement par le biais de l'API « Hypercarte.Data ». Les données sont alors sérialisées (faible occupation de la bande passante) et transmises avec l'applet lors d'une requête HTTP du client lourd. Le client charge en local les données et son interface réalisée en Swing par l'équipe du laboratoire LSR-IMAG. Il désérialise les données et

[41] Une définition des patrons de conception (ou modèles de conception) est donnée dans le catalogue de Gamma [GAMMA95] :
"Un patron de conception donne un nom, isole et identifie les principes fondamentaux d'une structure générale, pour en faire un moyen utile à l'élaboration d'une conception orientée objet réutilisable".

affiche la carte interactive. Le client accède aux données autonomes via le package "Hypercarte.Data".

III.2 Modélisation avec UML

III.2.1 Introduction

UML [MULLE97] [UML] pour Unified Modeling Langage est un langage de modélisation apparu en 1996 puis normalisé en 1997 (UML 1.1). Il est approuvé par le comité technique de l'OMG (Object Management Group) et supporté par les plus grandes sociétés d'informatique qui en font un standard de fait. UML 1.x propose plusieurs types de diagrammes pour la modélisation des aspects statiques et dynamiques du Système d'Information.
La démarche de modélisation retenue dans le cadre du module ATM est introduite à partir des cas d'utilisation pour conduire à l'état final retenu conformément à la figure 46.
Les figures présentées dans les sections suivantes ont été réalisées avec l'Atelier de Génie Logiciel (AGL) Rose de la société Rational [ROSE].

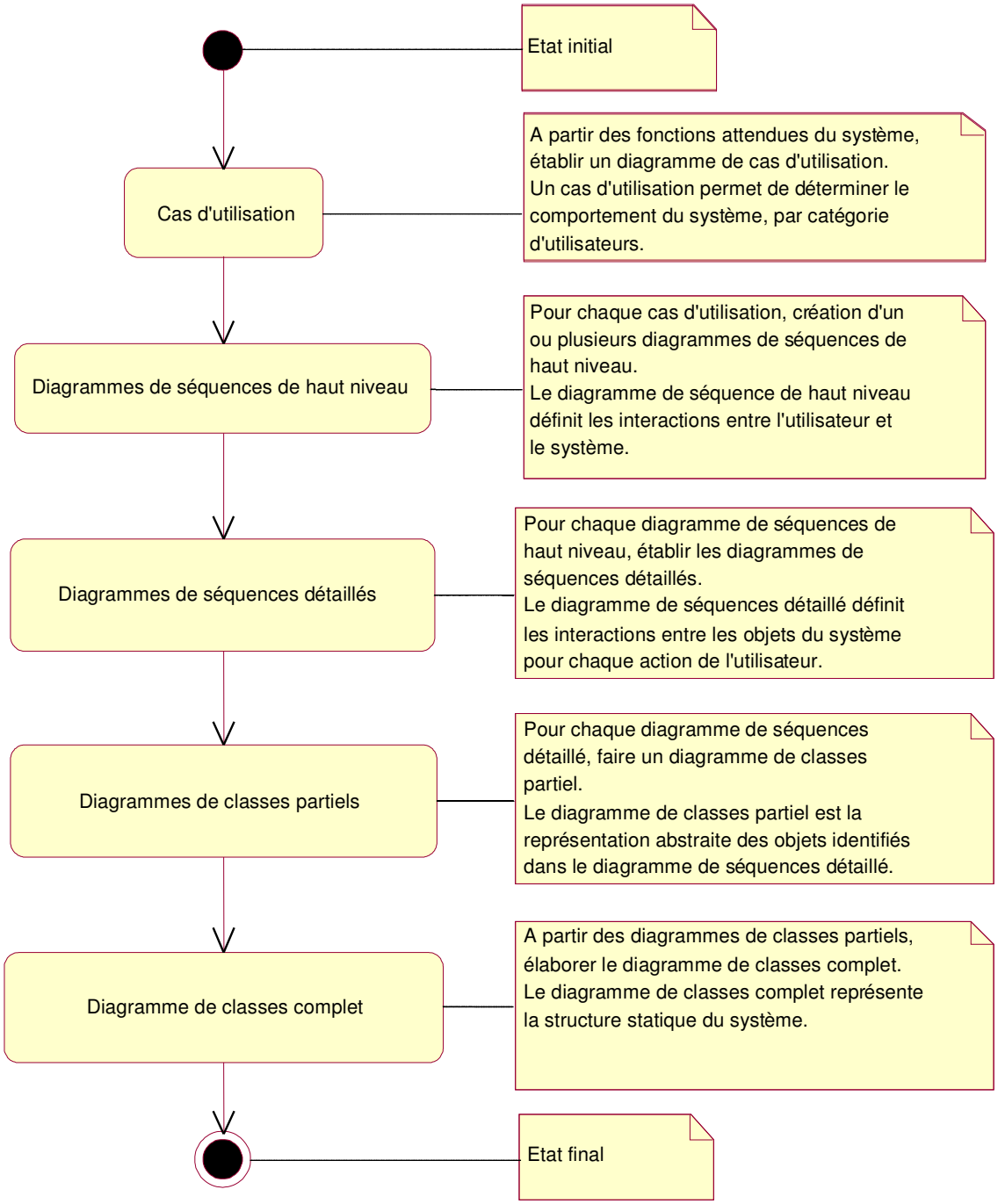


Figure 46 : La démarche de modélisation utilisée.

III.2.2 Cas d'utilisation et diagrammes

Les cas d'utilisation peuvent être observés suivant la manière dont on peut interagir avec le système par catégorie d'utilisateurs. Dans notre projet, exceptée la phase d'initialisation réalisée par le système, tous les cas d'utilisation concernent l'utilisateur de l'application.

Les retours des différents prototypes évalués successivement ont permis notamment de préciser le vocabulaire utilisé et les différentes fonctionnalités attendues par le système afin de clarifier et de rendre plus ergonomique l'interface utilisateur. Cette étude a été initialisée conjointement à partir de la maquette précisée (figure 2) et des spécifications cartographiques (annexes V.2 et V.3).

Les principales interactions attendues par le système avec l'utilisateur sont tout d'abord l'affichage des différentes cartes proposées, à savoir :

- la carte représentant l'espace et les maillages (carte des espaces),
- les cartes représentant les variables statistiques au nombre de 2 (carte des stocks 1 et 2)
- la carte représentant le ratio des 2 variables statistiques (carte de ratio)
- les cartes représentant les *déviations* (e.g. comparaison) hiérarchiques au nombre de 2 (carte de l'espace de référence et carte du découpage de référence)
- la carte représentant la déviation locale (carte de voisinage de référence)
- et la carte représentant la synthèse des déviations (carte de synthèse)

Les autres interactions attendues sont des fonctionnalités permettant d'interagir avec les différentes cartes affichées. Elles sont synthétisées et matérialisées par de grandes entités dans un souci de clarification pour le lecteur.

Une carte (au sens large[42]) est constituée :

- d'un cadre pour l'éditeur d'options regroupant différents choix utilisateurs :
    - à partir de listes déroulantes :
        - sélection d'un espace d'étude
        - sélection d'un découpage élémentaire
        - sélection d'un espace de référence ou (exclusif) d'une valeur de référence
        - sélection d'un découpage de référence
        - sélection d'un type de voisinage de référence
        - sélection du premier indicateur
        - sélection du deuxième indicateur
    - à partir de boutons pour :
        - augmenter le zoom
        - diminuer le zoom
        - permettre (oui/non) de déplacer la carte par déplacement de la souris (à partir du cadre de la carte)
        - permettre (oui/non) d'obtenir des informations contextuelles sur la carte à l'endroit sélectionné

[42] La carte identifie sa représentation cartographique, mais aussi sa légende ainsi que tous les éléments pour la constituer (e.g. les options).

- permettre (oui/non) d'obtenir un histogramme de synthèse sur la carte à l'endroit sélectionné

- d'un cadre pour la carte (représentation cartographique) regroupant trois possibilités pour l'utilisateur :
  - déplacer la carte par déplacement de la souris (« mouse_pressed ») (si le bouton « déplacer » est à l'état sélectionné)
  - obtenir des informations contextuelles sur la carte à l'endroit sélectionné par « survol » de la souris (si le bouton information est à l'état sélectionné)
  - obtenir un histogramme de synthèse sur la carte à l'endroit sélectionné par un « clic » de souris (si le bouton histogramme est à l'état sélectionné)

- d'un cadre pour l'« éditeur de légende » regroupant différents choix utilisateurs :
  - sélection d'une discrétisation (nombre de classes)
  - sélection d'un type de progression (arithmétique ou géométrique)
  - sélection d'une couleur pour chaque cercle à partir des couleurs pré-établies
  - sélection d'une palette de gamme de couleur simple pour la carte de ratio
  - sélection d'une palette de gamme de couleur double pour les cartes de déviations

Ces différents cas d'utilisation permettent d'établir le diagramme des cas d'utilisation présenté ci-dessous.

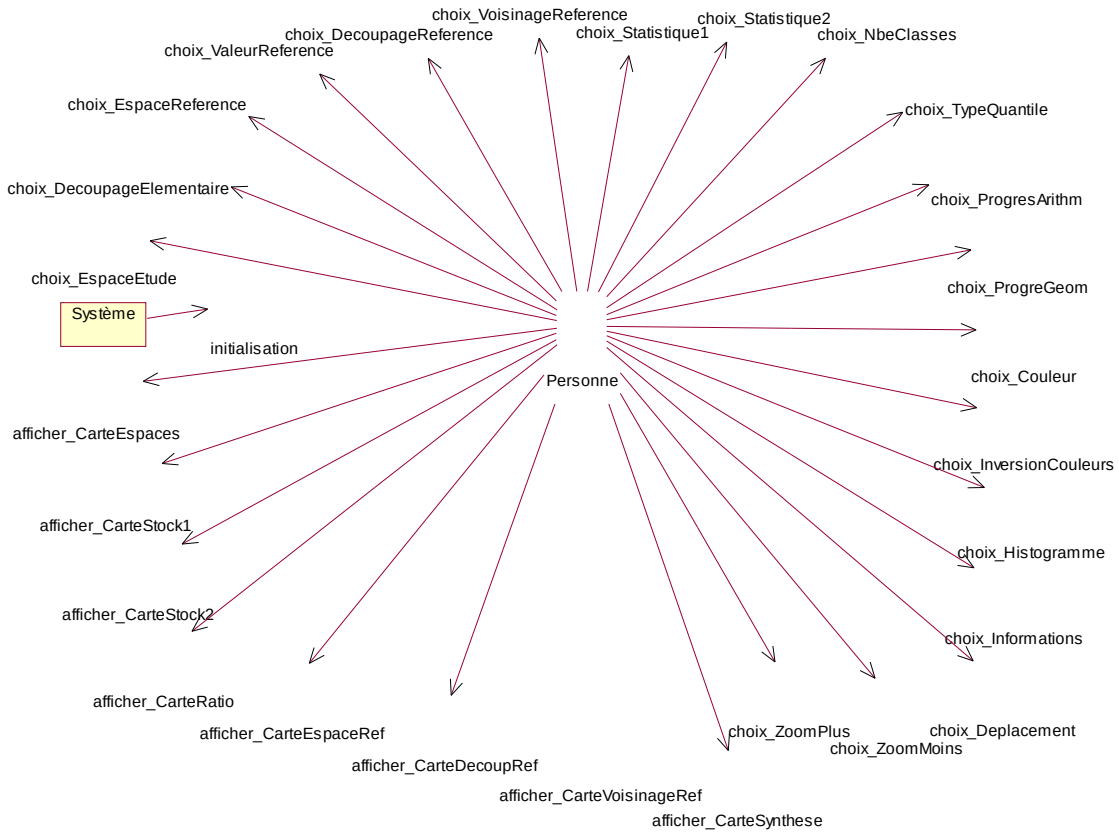


Figure 47 : Diagramme de cas d'utilisation.

Les différents cas d'utilisation utilisateur se regroupent en deux grands sous-ensembles : l'affichage des différentes cartes et les fonctionnalités issues des choix exprimés. Le système,

quant à lui, est responsable de l'initialisation des différents paramètres pour le bon fonctionnement de l'application.

Tous les cas d'utilisation ne pouvant être décrits, les choix retenus concernent :

- le cas d'utilisation initialisation réalisé par le système. Il permet la création des différents objets et leurs initialisations pour être « exploités » dans les cas d'utilisation suivants.
- le cas d'utilisation : choix_CartesEspaces (e.g. afficher_CarteEspaces)
- le cas choix_CarteStock1 (e.g. afficher_CarteStock1)
- le cas choix_EspaceReférence (e.g. afficher_CarteEspaceRef)

Les diagrammes de séquences de hauts niveaux ne seront pas directement exploités. En effet, dans ce type d'application ce sont les mécanismes internes au système qui sont intéressants. Ces diagrammes correspondent généralement à un choix utilisateur puis aux différents traitements qui conduisent à afficher une carte. Les réponses générales attendues du système seront néanmoins précisées à partir d'une petite description générale.

III.2.3 Cas d'utilisation système : Initialisation

Diagramme de séquences détaillé : initialisation (figure 48)

Description générale :
La phase d'initialisation permet, d'une part la création des différents objets généraux du système, d'autre part l'initialisation des objets à partir de données accessibles après la phase de désérialisation. Cet objectif sera complété par l'affichage de la carte des espaces pour la présenter à l'utilisateur comme carte par défaut.

Au chargement de l'application, le système informatique (ici le module d'analyse territoriale multiscalaire *Matm*) créé les principaux objets de notre application :

- l'objet de type Onglet permet de choisir une carte parmi celles disponibles. On doit le voir à ce stade de la modélisation comme une simple liste de choix affichée dans un formulaire.
- l'objet de type EditeurOptions qui propose des choix d'espaces, de statistiques, etc., à l'utilisateur.
- l'objet de type EditeurLegende qui propose des choix de couleurs, du nombre de classes, etc., pour certains types de cartes.
- l'objet de type ImageCarte qui affiche une représentation graphique (e.g. la carte).
- L'objet de type Legende qui créé 4 objets :
    - un objet de type ListeIntervalCouleurs utilisé pour la conception des aplats de couleurs.
    - un objet de type CalculCercle responsable du calcul des diamètres des cercles à afficher.
    - un objet de type CerclesLegende pour une représentation graphique des cercles sur la légende.
    - un objet de type Echelle pour une échelle de représentation graphique.

Le système désérialise ensuite les données afin de rendre disponible l'objet *ad* qui contient les différentes méthodes pour accéder aux données. Celles-ci sont récupérées (séquences 11 à 17) puis initialise l'objet *eo* (séquences 18 à 24). Un ensemble de type *PaletteCouleurs* est créé (25) à

partir des données recueillies en 17 puis rajouté et lié à *el*. L'objet *ic* est ensuite initialisé (séquences 28 à 35) ainsi que la légende *l* et son echelle *e* (séquences 36 à 41).
La séquence 42 réalise les objectifs du cas d'utilisation *afficher_CarteEspaces* décrit dans la suite du document.

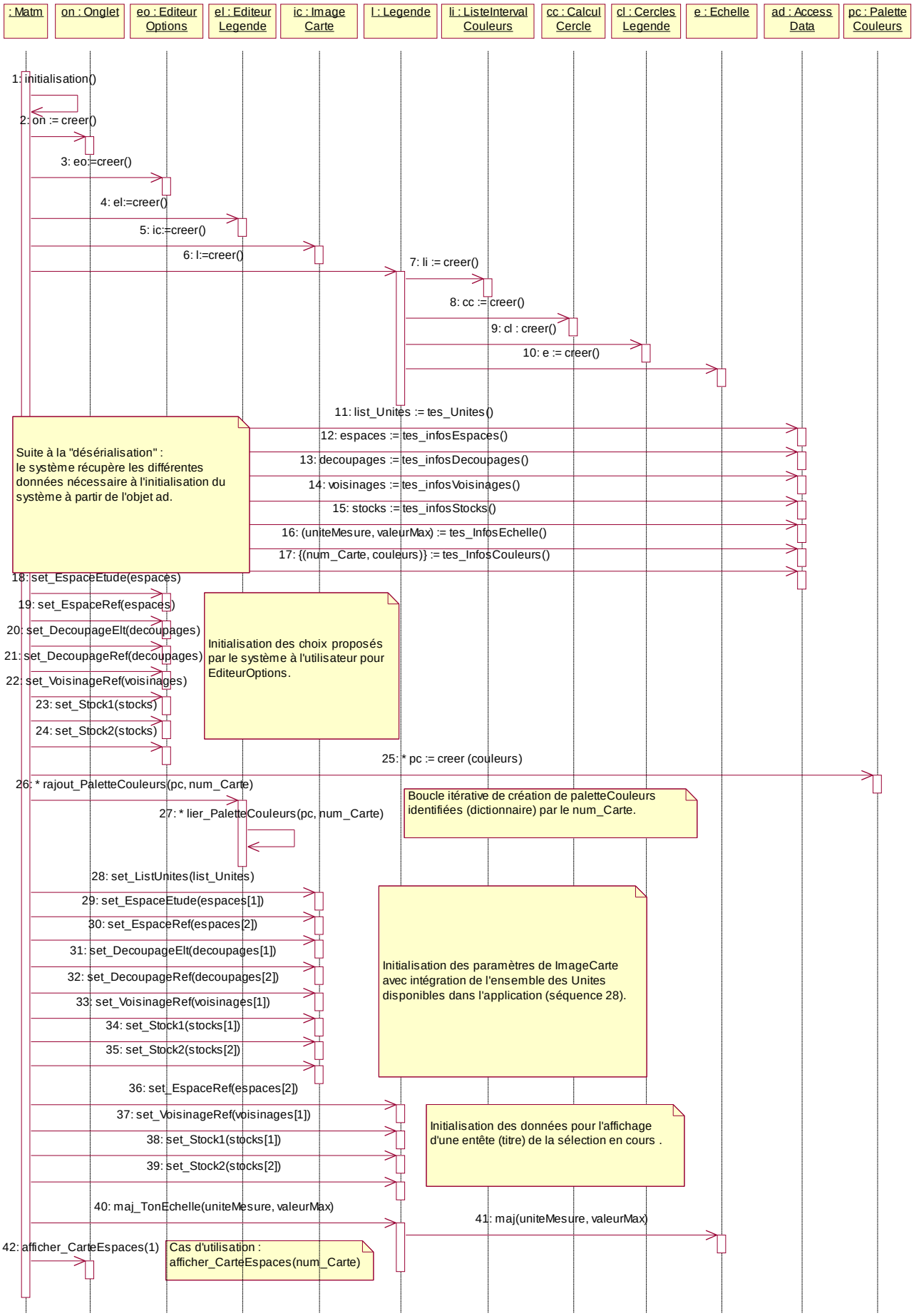
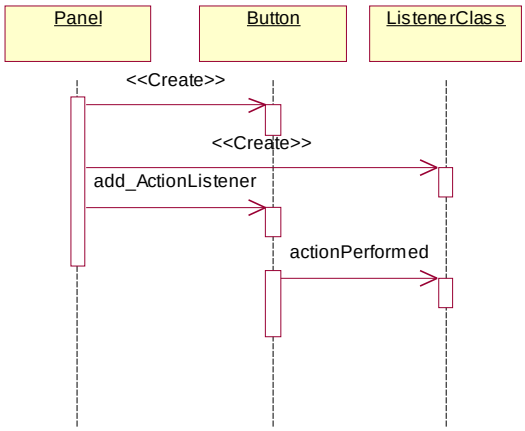


Figure 48 : Diagramme de séquences détaillé : Initialisation

Le cas d'utilisation initialisation permet, entre autres, de décrire certaines relations qui existent entre les objets du système. Par exemple, l'objet de type *Legende* est constitué de 4 objets, *li (ListeIntervalCouleurs)*, *cc (CalculCercle), cl (CerclesLegende)* et *e (Echelle)*. Il met en évidence la relation qui existe entre les différentes classes et ainsi la manière d'y accéder pour les cas d'utilisation suivants.
Remarque : le diagramme de classes partiel de ce cas d'utilisation n'est pas décrit.

III.2.4 Cas d'utilisation : Afficher_CarteEspaces

Remarques préalables :
Le module à développer réalise des interactions entre ses différents objets grâce à des évènements générés puis utilisés grâce aux méthodes associées dans son « listener ». L'exemple ci-dessous présente une utilisation classique. Un container (Panel) contient un bouton graphique, un « clic » de souris sur celui-ci entraîne la création d'un évènement (non visible sur la figure) exploité par la méthode actionPerformed de l'objet listener associé à l'évènement. L'objet qui hérite de l'objet listener doit implémenter le code de l'action à effectuer.



Dans notre étude, afin de ne pas surcharger les diagrammes de séquences, les boutons ainsi que les classes listener ne sont pas représentés. La figure ci-dessous exprime le fait qu'une action exécutée, par l'intermédiaire d'un bouton positionné sur la Classe1 ajoute un évènement qui sera écouté par la Classe2. Cette modélisation sera retenue pour les diagrammes de séquences.

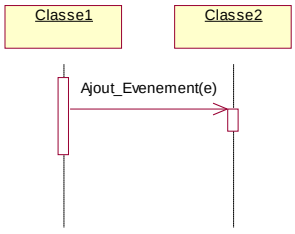


Diagramme de séquences détaillé : Afficher_CarteEspaces (figure 49)

Description générale :
L'utilisateur peut sélectionner, par le biais d'une liste d'onglets disponible sur l'interface, une carte des espaces. La carte des espaces permet de représenter l'espace de travail et les découpages que l'on souhaite utiliser pour notre analyse. Les unités appartenant à cet espace de travail (e.g. l'espace d'étude) sont affichées[43] *soit avec un contour fin de couleur quand elles appartiennent au découpage élémentaire, soit avec un contour épais de couleur quand elles appartiennent au découpage de référence. La surface des unités qui n'appartiennent pas à cet*

[43] Seul le contour est affiché.

espace de travail est quant à elle dessinée en gris. Le panneau graphique « EditeurLegende », situé à droite de la carte ne comporte aucun composant graphique, tandis que le panneau « Legende », situé à gauche contient une échelle graphique.

Cette carte est la première affichée par le système après initialisation de l'application. Mais il est possible d'y accéder par la suite grâce au cas d'utilisation : afficher_CarteEspaces().

Remarque : les séquences 1 et 2 simulent l'utilisation d'un formulaire dans lequel l'utilisateur saisirait une valeur (num_Carte). Ce n'est généralement pas le cas, mais le formalisme UML ne doit pas, à ce stade de la modélisation présumer des choix/possibilités du langage (e.g. utilisation d'onglets).

L'utilisateur sélectionne la carte des espaces (2) à partir du formulaire simulé par la liste d'onglets (1).
Le paramètre *num_carte* indique qu'il s'agit d'une carte des espaces.

> *Cette sélection créé un évènement (3) qui sera « écouté » par la classe EditeurLegende (4) précisant le type de carte sélectionné (num_Carte).*

L'éditeur de légende utilise la méthode *choixCarte* associée à l'événement *cce (6)*. L'événement traité permet d'identifier une carte des espaces.

> *Le même évènement ChoixCarteEvent est transmis (7) et écouté par la classe ImageCarte (précisant toujours le type de carte sélectionné)*

En (9), l'événement *cce* est traité et la méthode reconnaît une carte des espaces. Le traitement « afficher_CarteEspaces[44] » (séquences 10 à 13) peut avoir lieu sur l'ensemble des unités disponibles dans l'application.

> *Le même évènement ChoixCarteEvent sera transmis (14) et écouté par la classe Légende (précisant toujours le type de carte sélectionné)*

La légende traite l'événement en (16) puis créé un autre événement en (17) qui sera transmis à la classe *Matm* (18). Celui-ci le traite grâce à la méthode *validation* entraînant un affichage simultané de tous les composants graphique de l'application à l'utilisateur (20).

Remarque : l'ordonnancement d'envoi d'évènements ne garantit pas le traitement de ceux-ci dans le même ordre. C'est pourquoi, afin de respecter l'ordre d'arrivée des différentes modifications nécessaires à l'application, on choisit de constituer des évènements de type transitoire. Un évènement traité retransmet ce même évènement qui sera traité, etc.
Cette organisation est retenue pour l'ensemble des cartes. Chaque objet parcouru réalise des traitements sur ses composants (affichage/traitement ou non) grâce à la méthode *choixCarte(ChoixCarteEvent)* qui reconnaît la carte à traiter.

[44] Cette méthode appliquée à l'objet imageCarte n'est pas représentée dans le diagramme.

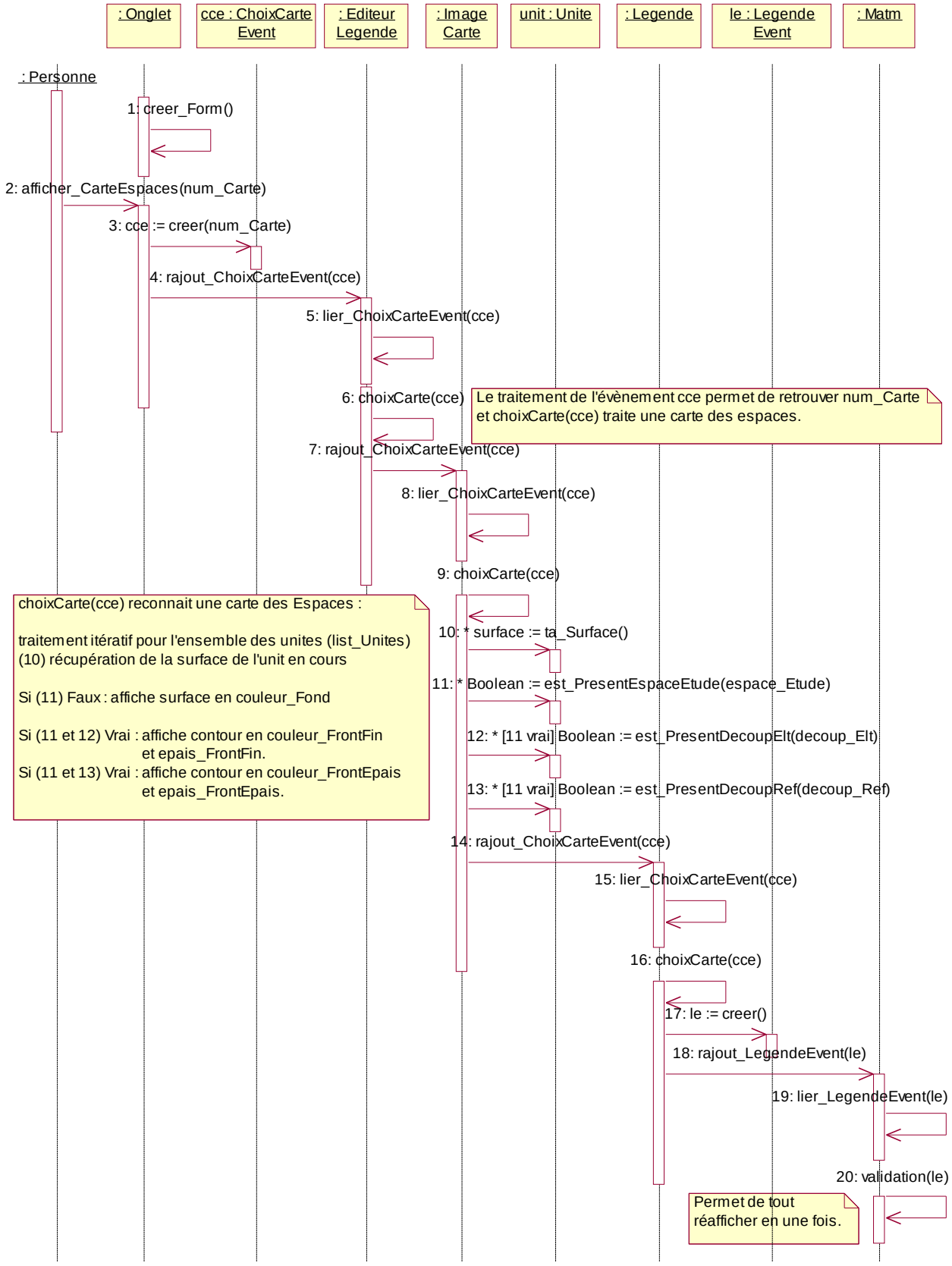


Figure 49 : Diagramme de séquences détaillé : Afficher_CarteEspaces

Afin de ne pas surcharger les diagrammes de séquences, les séquences 17 à 20 seront omises dans les prochaines descriptions. Elles sont cependant toujours utilisées.

Diagramme de classes partiel : afficher_CarteEspaces (figure 50)

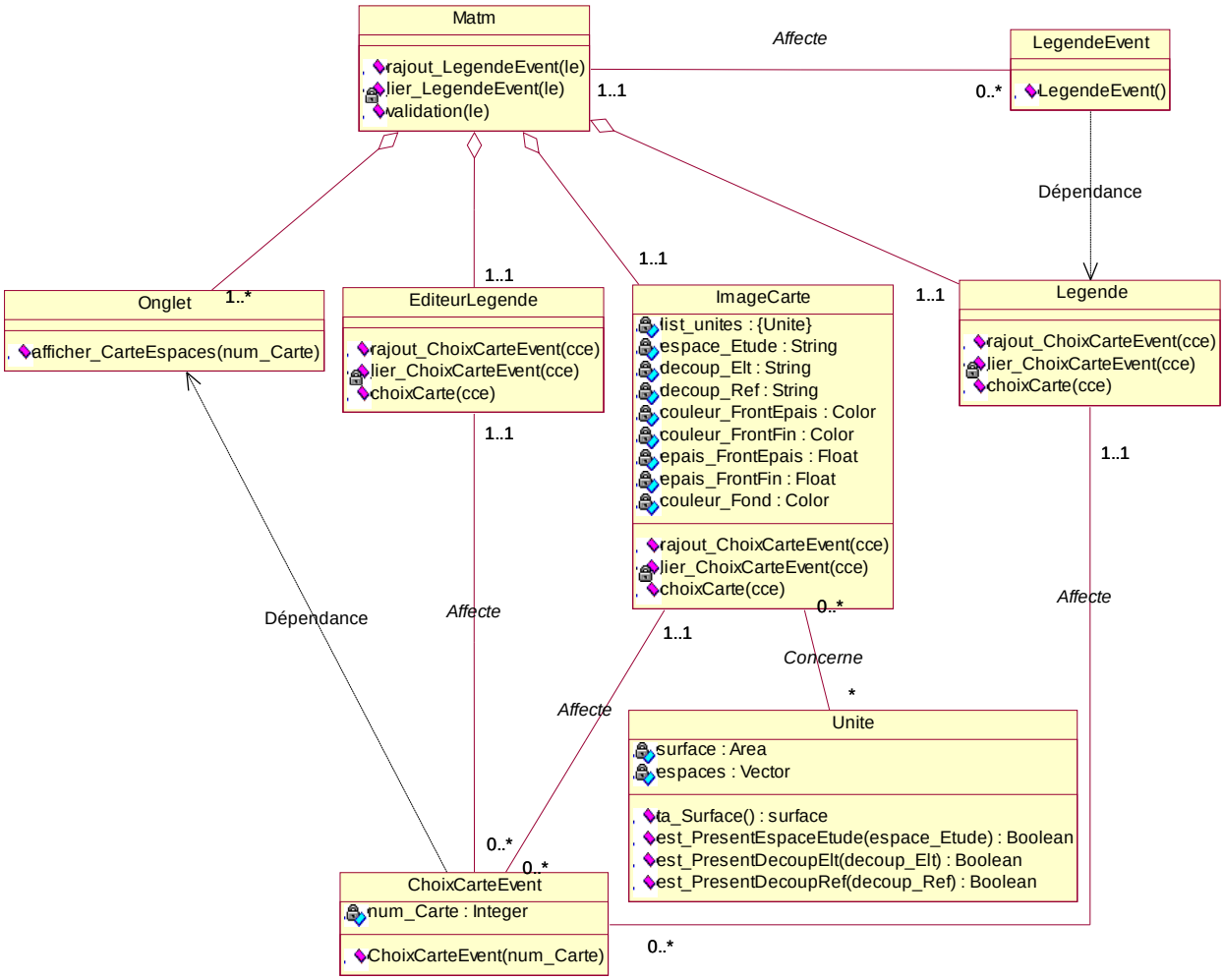


Figure 50 : Diagramme de classes partiel : Afficher_CarteEspaces

- L'événement de type *ChoixCarteEvent* dépend de l'utilisation de la classe Onglet. Il sera « écouté » par les classes *EditeurLegende, ImageCarte* et *Legende* puis exploité grâce à la méthode *choixCarte(ChoixCarteEvent)* présente dans les 3 classes. L'événement *LegendeEvent* produit à partir de la classe *Legende* sera utilisé via sa méthode *validation* dans la classe *Matm*.

- Les attributs d'espace, de découpage, de couleur et d'épaisseur des frontières de la classe *ImageCarte* sont déduits pour l'affichage de la carte des espaces (non représentés dans le diagramme de séquences).

- Les méthodes de création comme la séquence 3 (cce : creer(num_Carte)) appliquées à l'objet de type *ChoixCarteEvent* sont remplacées dans le diagramme de classes partiel par le nom du constructeur (ChoixCarteEvent(num_Carte)).

- Les évènements peuvent posséder un attribut. C'est le cas pour *ChoixCarteEvent(num_Carte)* quand il apporte une information, ici le numéro de la carte. Cependant l'utilisation de l'événement de type *LegendeEvent* sans paramètre est suffisant pour signaler à l'objet *Matm* de valider la transaction.

III.2.5 Cas d'utilisation : Afficher_CarteStock1

Diagramme de séquences détaillé : Afficher_CarteStock1 (figure 51)

Description générale :
L'utilisateur peut sélectionner, par le biais d'une liste d'onglets disponible sur l'interface, une carte représentant une variable statistique. La représentation de cette carte, que l'on nommera carte de stock, se fait à l'aide de cercles proportionnels par leur surface à la valeur de l'attribut à représenter. Les frontières sont aussi affichées. Les cercles sont pleins et prennent la couleur correspondant à la dernière sélection effectuée pour cette carte. Les cercles du plus grand et du plus petit diamètre sont identiques quelque soit l'indicateur utilisé. Deux cartes peuvent être représentées suivant le choix de la valeur statistique à traiter. Dans cette étude, le premier indicateur est utilisé (e.g. stock1). Le panneau graphique « EditeurLegende », situé à droite de la carte affiche la couleur utilisée par les cercles. Le panneau « Legende », situé à gauche de la carte contient une échelle graphique et 3 cercles représentant les valeurs minimale, maximale et intermédiaire.

- Le traitement de l'événement *ChoixCarteEvent* se réalise comme précédemment. Il permet, grâce aux méthodes *choixCarte(ChoixCarteEvent),* implémentées dans les classes principales, de reconnaître une carte de stock.

- En (7), la méthode *ta_Palette* en fonction du paramètre *num_Carte* permet d'identifier l'objet *pc*. La méthode *ta_Couleur(indice_Stock1)* est soumise à l'objet *pc* (8) qui lui renvoie la couleur pour l'affichage ultérieur de ses cercles.

- La séquence (9) permet d'afficher la couleur en cours qui correspond à la dernière sélection effectuée sur cette carte. Le traitement graphique offre à l'utilisateur le choix d'une autre couleur parmi une palette (de couleurs) disponible.

- Les unités que l'on représente appartiennent à un espace d'étude et à un découpage élémentaire (14). La valeur statistique prise en considération est le *stock* (e.g. valeur) donné par le premier indicateur statistique (16).

- En (15), les valeurs minimale et maximale sont calculées en fonction des valeurs statistiques des unités appartenant à l'espace considéré. Elles permettent de mettre à jour les valeurs *min* et *max* dans l'objet *cc (CalculCecle)* (17). Elles seront utilisées ultérieurement pour calculer le diamètre des cercles (20).

- La séquence 25 correspond au traitement préalable pour l'affichage graphique des 3 cercles.

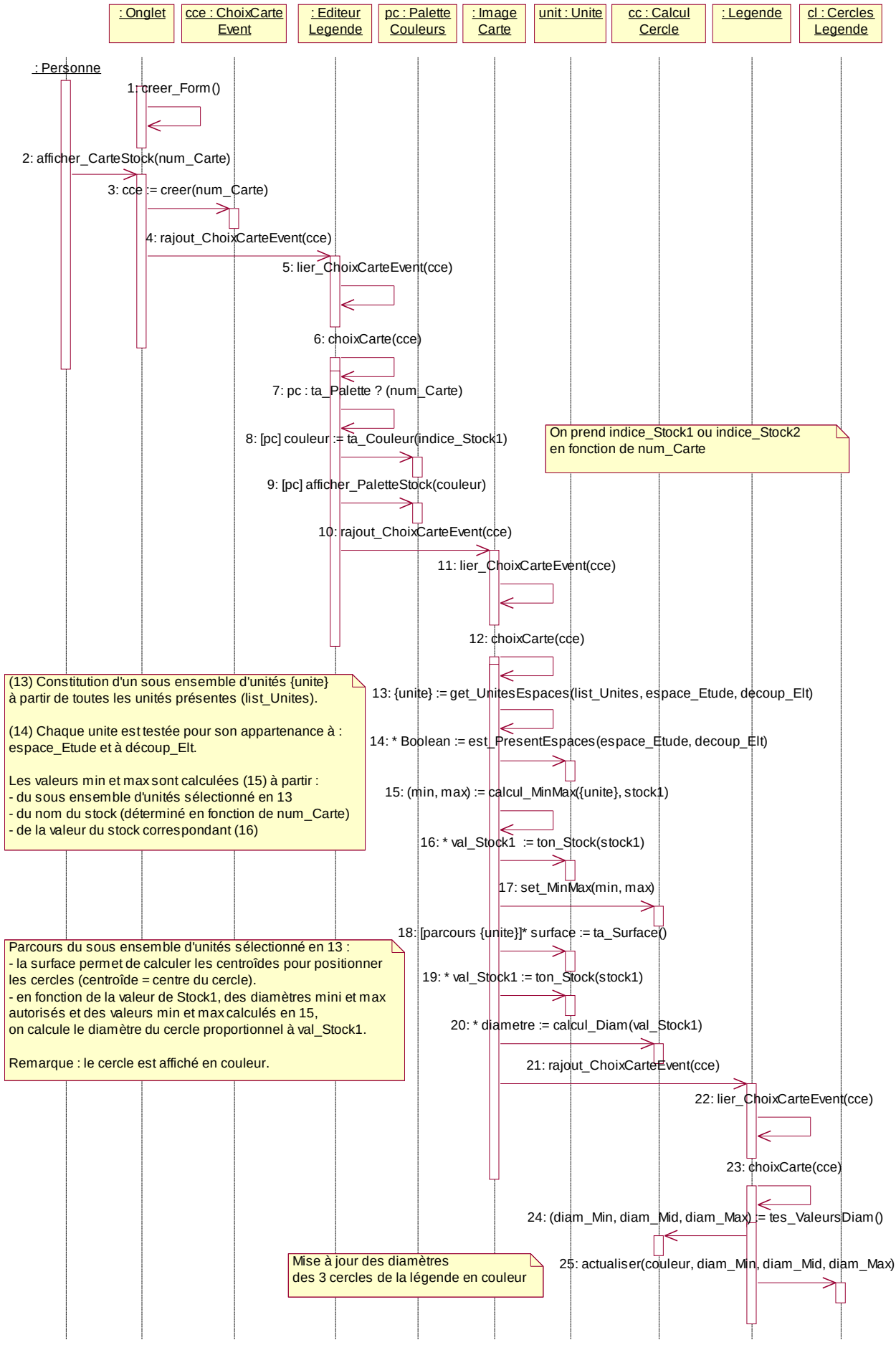

: Onglet
cce : ChoixCarte Event
: Editeur Legende
pc : Palette Couleurs
: Image Carte
unit : Unite
cc : Calcul Cercle
: Legende
cl : Cercles Legende
: Personne
1: creer_Form()
2: afficher_CarteStock(num_Carte)
3: cce := creer(num_Carte)
4: rajout_ChoixCarteEvent(cce)
5: lier_ChoixCarteEvent(cce)
6: choixCarte(cce)
7: pc : ta_Palette ? (num_Carte)
8: [pc] couleur := ta_Couleur(indice_Stock1)
On prend indice_Stock1 ou indice_Stock2
en fonction de num_Carte
9: [pc] afficher_PaletteStock(couleur)
10: rajout_ChoixCarteEvent(cce)
11: lier_ChoixCarteEvent(cce)
12: choixCarte(cce)
(13) Constitution d'un sous ensemble d'unités {unite}
à partir de toutes les unités présentes (list_Unites).
(14) Chaque unite est testée pour son appartenance à :
espace_Etude et à découp_Elt.
Les valeurs min et max sont calculées (15) à partir :
- du sous ensemble d'unités sélectionné en 13
- du nom du stock (déterminé en fonction de num_Carte)
- de la valeur du stock correspondant (16)
13: {unite} := get_UnitesEspaces(list_Unites, espace_Etude, decoup_Elt)
14: * Boolean := est_PresentEspaces(espace_Etude, decoup_Elt)
15: (min, max) := calcul_MinMax({unite}, stock1)
16: * val_Stock1 := ton_Stock(stock1)
17: set_MinMax(min, max)
18: [parcours {unite}]* surface := ta_Surface()
Parcours du sous ensemble d'unités sélectionné en 13 :
- la surface permet de calculer les centroïdes pour positionner
les cercles (centroïde = centre du cercle).
- en fonction de la valeur de Stock1, des diamètres mini et max
autorisés et des valeurs min et max calculés en 15,
on calcule le diamètre du cercle proportionnel à val_Stock1.
Remarque : le cercle est affiché en couleur.
19: * val_Stock1 := ton_Stock(stock1)
20: * diametre := calcul_Diam(val_Stock1)
21: rajout_ChoixCarteEvent(cce)
22: lier_ChoixCarteEvent(cce)
23: choixCarte(cce)
24: (diam_Min, diam_Mid, diam_Max) := tes_ValeursDiam()
Mise à jour des diamètres
des 3 cercles de la légende en couleur
25: actualiser(couleur, diam_Min, diam_Mid, diam_Max)

Figure 51 : Diagramme de séquences détaillé : Afficher_CarteStock1

Diagramme de classes partiel : afficher_CarteStock1 (figure 52)

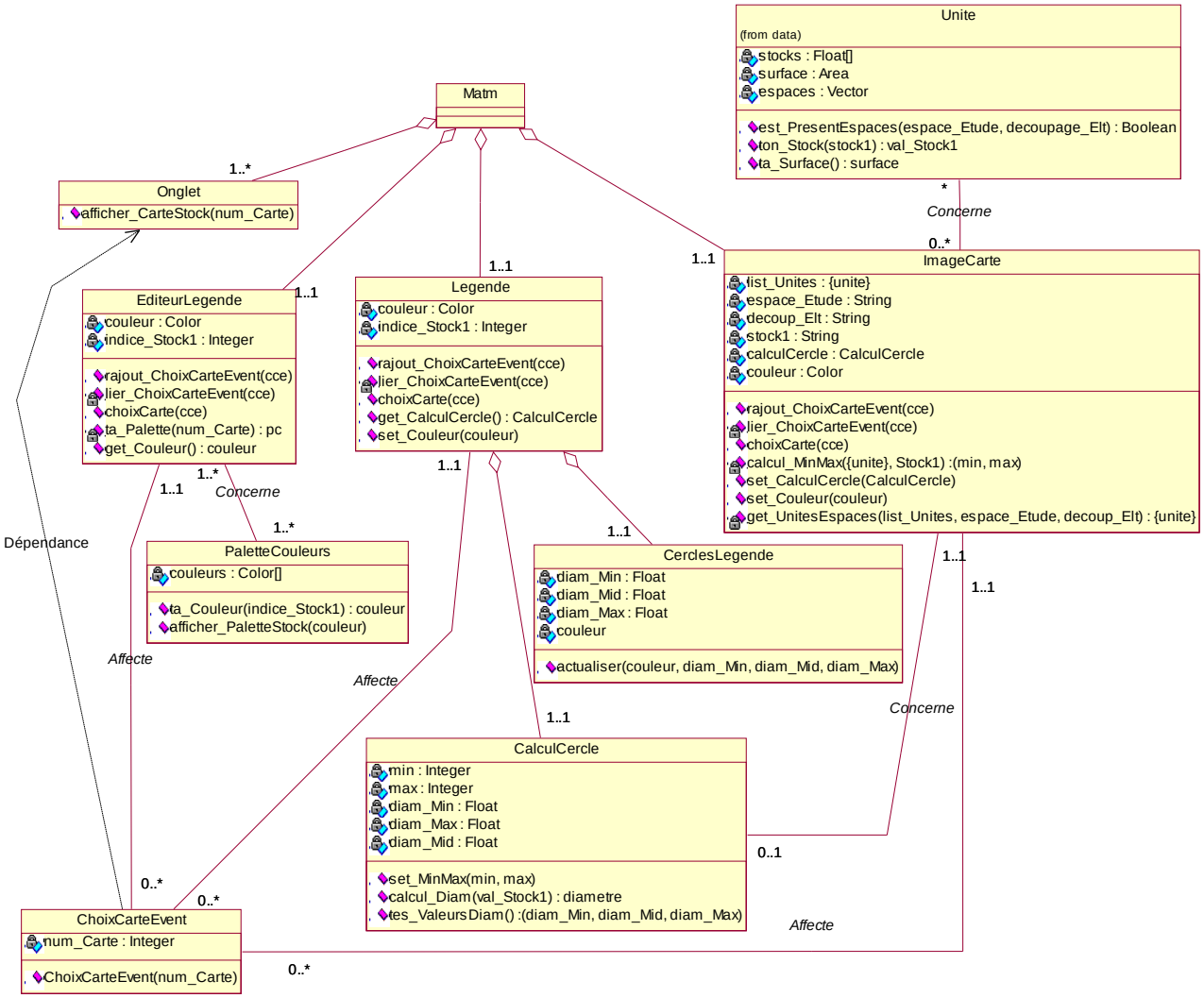


Figure 52 : Diagramme de classes partiel : Afficher_CarteStock1

Certains attributs sont partagés entre les classes par le système :

- L'objet de la classe *ImageCarte* connaît l'objet *cc* (CalculCercle) par l'intermédiaire de la classe *Matm* (e.g. le système). Les méthodes *get_CalculCercle()* de la classe *Legende* et *set_CalculCercle(CalculCercle)* de la classe *ImageCarte* permettent ces accès.

- De même, l'attribut couleur utilisé pour l'affichage des cercles dans *ImageCarte* et dans CerclesLegende est accessible grâce aux méthodes *get_Couleur()* dans *EditeurLegende* et *set_Couleur(couleur)* dans *ImageCarte et Legende*.

III.2.6 Cas d'utilisation : Afficher_CarteEspaceRef

Diagramme de séquences détaillé : Afficher_CarteEspaceRef (figure 53)

Description générale :
L'utilisateur peut sélectionner, par le biais d'une liste d'onglets disponible sur l'interface, une carte de déviations hiérarchiques (ici la carte EspaceRef). Elle est affichée à l'aide d'aplats (e.g. couleur pleine) dans une gamme de couleurs « double[45] ». La légende correspondante sera visualisée grâce à des « briques » de couleurs complétées de la fréquence[46] et de son intervalle (limite inférieure). Elle disposera également d'une échelle graphique. Le panneau graphique « EditeurLegende », situé à droite de la carte affiche la gamme de couleurs utilisée.
Le ratio désiré par l'utilisateur (e.g. stock1 / stock2) de chaque unité appartenant à l'espace d'étude et au découpage élémentaire sera comparé au ratio globalisé des unités appartenant à l'espace de référence (avec indice 100 = moyenne des unités de l'espace de référence).

- La séquence (9) récupère un ensemble de couleurs car, contrairement à une carte des stocks, celle-ci utilise une palette entière de couleurs.

- Les séquences (13 et 14) permettent de calculer l'indice de référence pour notre base 100.

- Les séquences (22 à 24) créent des intervalles de couleurs composés d'une limite inférieure, supérieure et d'une couleur qui seront utilisées dans la séquence 33. La valeur de delta est comparée aux limites inférieures et supérieures. Si l'intervalle englobe la valeur, la méthode retourne la couleur correspondante et la carte peut afficher la surface de l'unité.

[45] Deux dominantes de couleurs aux extrémités avec un certain nombre de couleurs intermédiaires.
[46] Nombre d'unités appartenant à l'intervalle délimité par une limite inférieure et une limite supérieure.

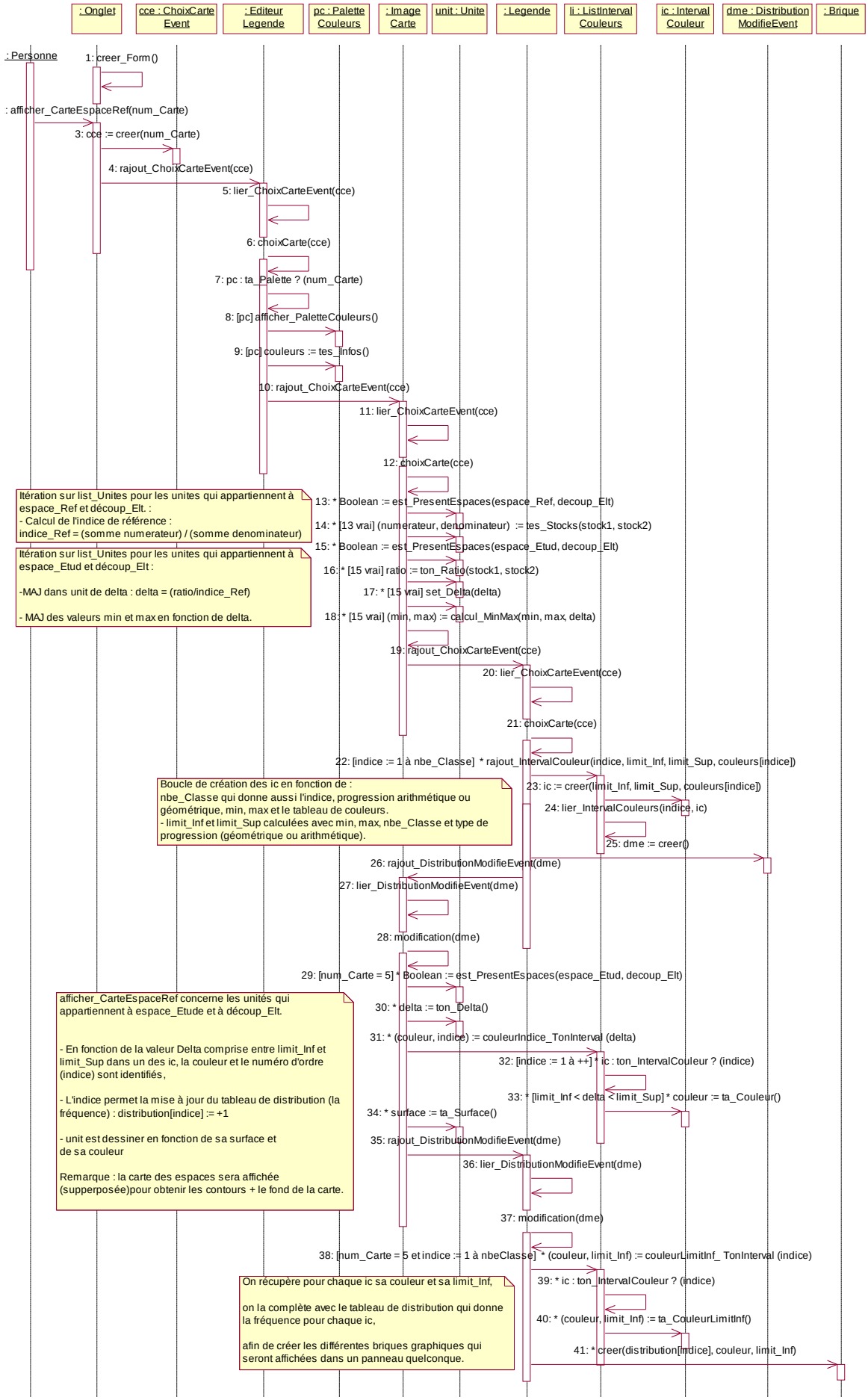

: Onglet
cce : ChoixCarte Event
: Editeur Legende
pc : Palette Couleurs
: Image Carte
unit : Unite
: Legende
li : ListInterval Couleurs
ic : Interval Couleur
dme : Distribution ModifieEvent
: Brique
: Personne
1: creer_Form()
: afficher_CarteEspaceRef(num_Carte)
3: cce := creer(num_Carte)
4: rajout_ChoixCarteEvent(cce)
5: lier_ChoixCarteEvent(cce)
6: choixCarte(cce)
7: pc : ta_Palette ? (num_Carte)
8: [pc] afficher_PaletteCouleurs()
9: [pc] couleurs := tes_Infos()
10: rajout_ChoixCarteEvent(cce)
11: lier_ChoixCarteEvent(cce)
12: choixCarte(cce)
Itération sur list_Unites pour les unites qui appartiennent à espace_Ref et découp_Elt. :
- Calcul de l'indice de référence :
indice_Ref = (somme numerateur) / (somme denominateur)
13: * Boolean := est_PresentEspaces(espace_Ref, decoup_Elt)
14: * [13 vrai] (numerateur, denominateur) := tes_Stocks(stock1, stock2)
Itération sur list_Unites pour les unites qui appartiennent à espace_Etud et découp_Elt :
-MAJ dans unit de delta : delta = (ratio/indice_Ref)
- MAJ des valeurs min et max en fonction de delta.
15: * Boolean := est_PresentEspaces(espace_Etud, decoup_Elt)
16: * [15 vrai] ratio := ton_Ratio(stock1, stock2)
17: * [15 vrai] set_Delta(delta)
18: * [15 vrai] (min, max) := calcul_MinMax(min, max, delta)
19: rajout_ChoixCarteEvent(cce)
20: lier_ChoixCarteEvent(cce)
21: choixCarte(cce)
22: [indice := 1 à nbe_Classe] * rajout_IntervalCouleur(indice, limit_Inf, limit_Sup, couleurs[indice])
Boucle de création des ic en fonction de :
nbe_Classe qui donne aussi l'indice, progression arithmétique ou géométrique, min, max et le tableau de couleurs.
- limit_Inf et limit_Sup calculées avec min, max, nbe_Classe et type de progression (géométrique ou arithmétique).
23: ic := creer(limit_Inf, limit_Sup, couleurs[indice])
24: lier_IntervalCouleurs(indice, ic)
25: dme := creer()
26: rajout_DistributionModifieEvent(dme)
27: lier_DistributionModifieEvent(dme)
28: modification(dme)
29: [num_Carte = 5] * Boolean := est_PresentEspaces(espace_Etud, decoup_Elt)
afficher_CarteEspaceRef concerne les unités qui appartiennent à espace_Etude et à découp_Elt.
- En fonction de la valeur Delta comprise entre limit_Inf et limit_Sup dans un des ic, la couleur et le numéro d'ordre (indice) sont identifiés,
- L'indice permet la mise à jour du tableau de distribution (la fréquence) : distribution[indice] := +1
- unit est dessiner en fonction de sa surface et de sa couleur
Remarque : la carte des espaces sera affichée (supperposée)pour obtenir les contours + le fond de la carte.
30: * delta := ton_Delta()
31: * (couleur, indice) := couleurIndice_TonInterval (delta)
32: [indice := 1 à ++] * ic : ton_IntervalCouleur ? (indice)
33: * [limit_Inf < delta < limit_Sup] * couleur := ta_Couleur()
34: * surface := ta_Surface()
35: rajout_DistributionModifieEvent(dme)
36: lier_DistributionModifieEvent(dme)
37: modification(dme)
38: [num_Carte = 5 et indice := 1 à nbeClasse] * (couleur, limit_Inf) := couleurLimitInf_ TonInterval (indice)
On récupère pour chaque ic sa couleur et sa limit_Inf,
on la complète avec le tableau de distribution qui donne la fréquence pour chaque ic,
afin de créer les différentes briques graphiques qui seront affichées dans un panneau quelconque.
39: * ic : ton_IntervalCouleur ? (indice)
40: * (couleur, limit_Inf) := ta_CouleurLimitInf()
41: * creer(distribution[indice], couleur, limit_Inf)

Figure 53 : Diagramme de séquences détaillé : Afficher_CarteEspaceRef

Diagramme de classes partiel : Afficher_CarteEspaceRef (figure 54)

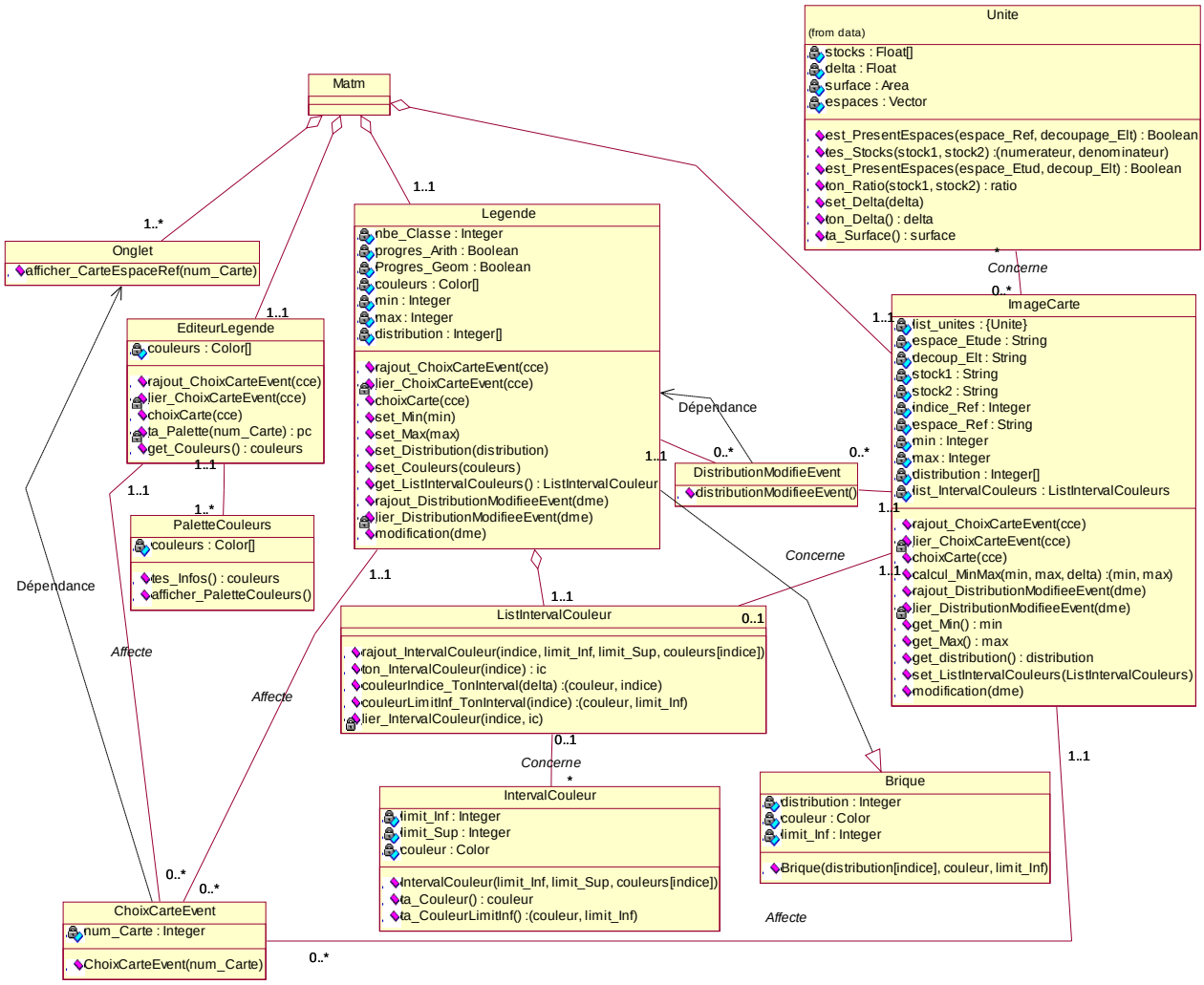


Figure 54 : Diagramme de classes partiel : Afficher_CarteEspaceRef

Certains attributs sont partagés entre les classes par le système :

- L'objet de la classe *Image_Carte* connaît l'objet *li* (ListIntervalCouleurs) par l'intermédiaire de la classe *Matm*. Les méthodes *get_ListIntervalCouleurs()* de la classe *Legende* et *set_ListIntervalCouleurs(ListIntervalCouleurs)* de la classe *ImageCarte* permettent ces accès.

- L'attribut couleurs de la classe *EditeurLegende* est utilisé avec la méthode *get_Couleurs()* pour être enregistré dans la classe *Legende* par *set_Couleurs(couleurs).* Cet ensemble de couleurs est responsable (en partie) de la création des briques (séquence 38).

- Trois autres attributs sont partagés, les valeurs *min*, *max* et le *tableau de distribution* sont créés par l'objet de type *ImageCarte* et transmis en lecture seul à l'objet de type *Legende* via l'objet *matm* suivant les méthodes *set* et *get* correspondantes.

III.2.7 Aperçu des évènements

Nous avons déjà observé 3 types d'évènements :

- ChoixCarteEvent utilisé pour reconnaître le type de carte en cours,
- LegendeEvent pour signaler au système que tout est réalisé, il peut tout afficher,
- DistributionModifieEvent pour procéder à des modifications successives entre les objets de type ImageCarte et Legende.

Il existe d'autres évènements qui ne peuvent bénéficier d'une modélisation aussi approfondie que précédemment. Cependant, on peut décrire succinctement qui est à l'origine de l'événement et les interactions provoquées avec les autres classes du système. Pour cela, on utilise une interface de type « listener » qui permet de décrire la méthode utilisée conjointement avec l'événement.

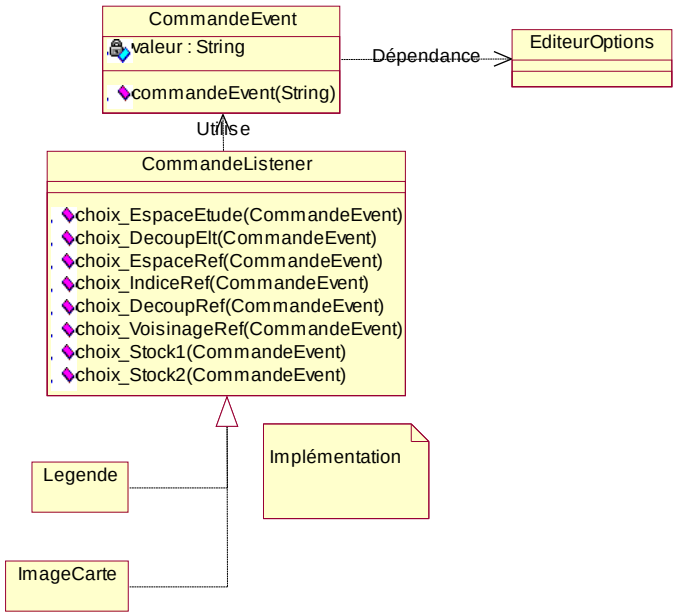
Les principales interfaces réalisées dans ce document concernent :

- Les interfaces CommandeListener, VisualisationListener et ZoomListener qui utilisent des évènements provoqués par un objet de la classe EditeurOptions.
- Les interfaces EditeurLegendeListener et CouleursListener qui dépendent des événements produits par les objets de type EditeurLegende et PaletteCouleurs respectivement.

Interface CommandeListener

La sélection par l'utilisateur d'un nouvel espace, d'un nouveau découpage ou d'un autre stock entraîne la création d'un événement *CommandeEvent* par l'objet de type EditeurOptions.
Cet événement est écouté par les classes suivantes qui réalisent les opérations en implémentant le code de la méthode :

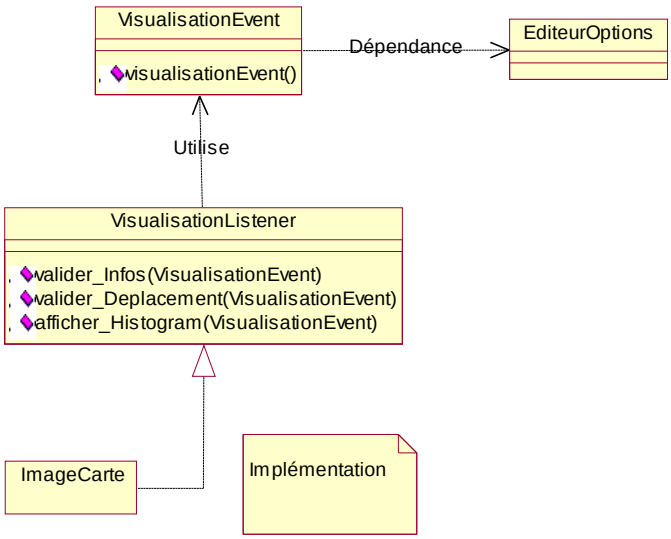
- Legende :
  - Modification du titre (nom du stock, indice de référence, etc.)
  - Modification du diamètre des cercles proportionnels aux nouvelles valeurs rencontrées (carte des stocks)
  - Ou modification de la distribution qui entraîne un changement pour les aplats de couleurs (autres cartes)
- ImageCarte :
  - La légende modifiée entraîne une nouvelle représentation de la carte.



Remarque : cet événement est porteur d'un attribut de type *String* pour la chaîne sélectionnée.

Interface VisualisationListener

L'utilisateur peut cliquer sur un bouton pour signaler à la carte (ImageCarte) qu'elle valide un futur déplacement (l'utilisateur pourra cliquer la carte en un point et relâcher sa souris en un autre provoquant une translation de la carte). L'utilisateur peut aussi désirer des informations contextuelles sur la carte qui seront aussi signalées à la carte. Elles peuvent prendre la forme d'informations textuelles par survol de la souris (bouton valider_Infos) ou d'histogramme par sélection d'une unité (bouton afficher_Histogram)

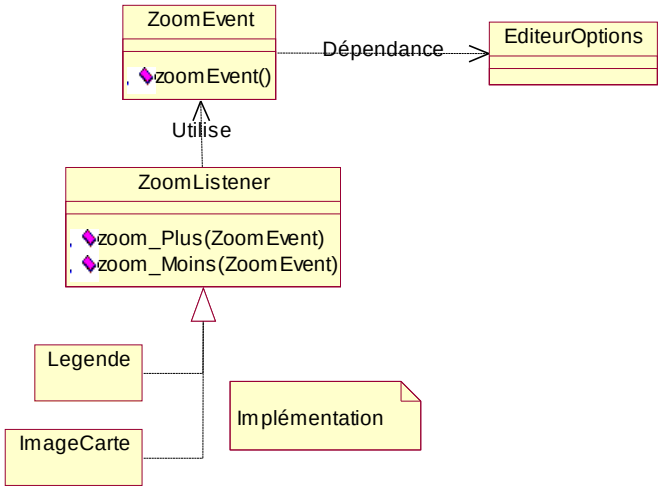


L'événement de type VisualisationEvent est créé par un objet de la classe EditeurOptions puis écouté par la seule classe ImageCarte qui met à jour ces indicateurs (valider_Infos Oui/Non etc.). Par la suite l'utilisateur pourra (effectivement) utiliser ces fonctionnalités directement sur la carte via sa souris.

Interface ZoomListener

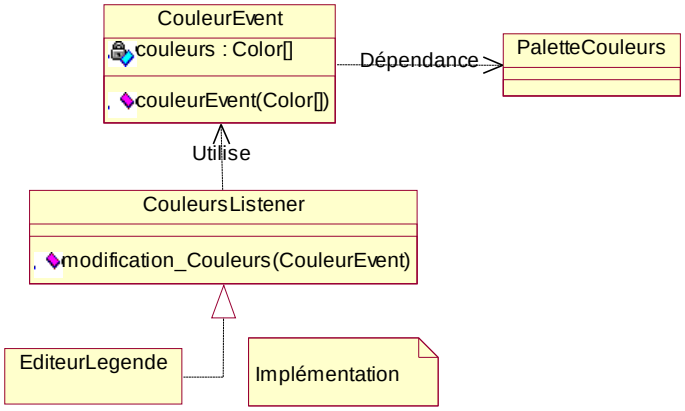
L'utilisateur peut cliquer sur le bouton Zoom_Plus ou Zoom_Moins dans l'éditeur des options. Dans ce cas, un changement est directement réalisé sur la carte (les frontières sont redessinées).

De plus, s'il s'agit d'une carte de stock, les cercles grandissent (e.g. zoom_Plus) proportionnellement jusqu'à une limite définie.
Le panneau CercleLegende et l'échelle graphique (e.g. echelle) sont aussi modifiés par l'intermédiaire de la classe Legende.



Interface CouleursListener

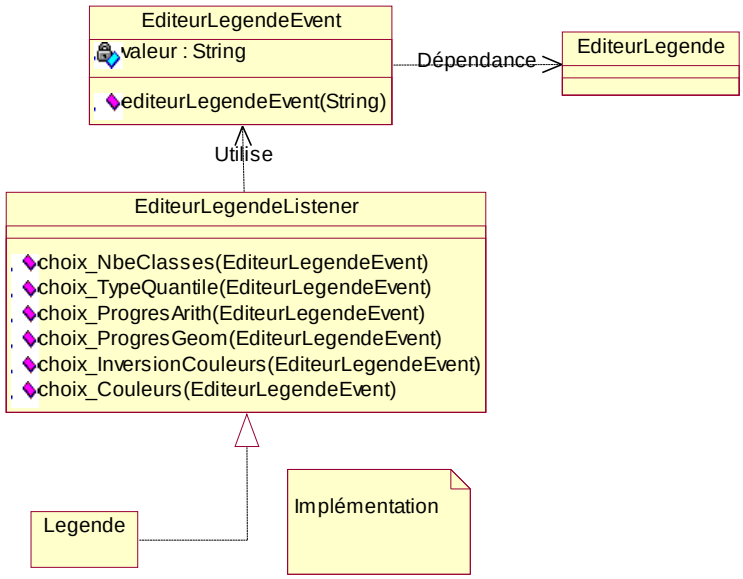
L'objet de type PaletteCouleurs permet d'offrir à l'utilisateur un choix de couleurs parmi celles proposées. Ce choix provoque la création d'un événement de type CouleursEvent qui sera écouté directement par la seule classe EditeurLegende avec comme paramètre une couleur pour une carte des stocks ou un ensemble de couleurs pour les autres cartes.



Cependant, il faut avertir les autres classes de la modification survenue. Un autre événement sera alors créé de type *EditeurLegendeEvent* qui sera exploité grâce à la méthode *choix_Couleurs* pour indiquer ce changement. Il est présenté ci-dessous.

Interface EditeurLegendeListener

Les choix utilisateurs effectués à partir *EditeurLegende* (choix du nombre de classes à utiliser, type de progression etc.) créé des évènements de type EditeurLegendeEvent qui sont écoutés par la seule classe Legende.



La modification des paramètres de Legende influe nécessairement sur ImageCarte. L'événement DistributionModifieEvent (déjà connu) permet de signaler ces changements.

III.2.8 Diagramme de classes complet

Deux diagrammes de classes sont décrits. Le premier (figure 55) présente le diagramme de classes complet sans propriété (attributs et méthodes). Le second diagramme (figure 56) intègre l'ensemble des attributs et méthodes.

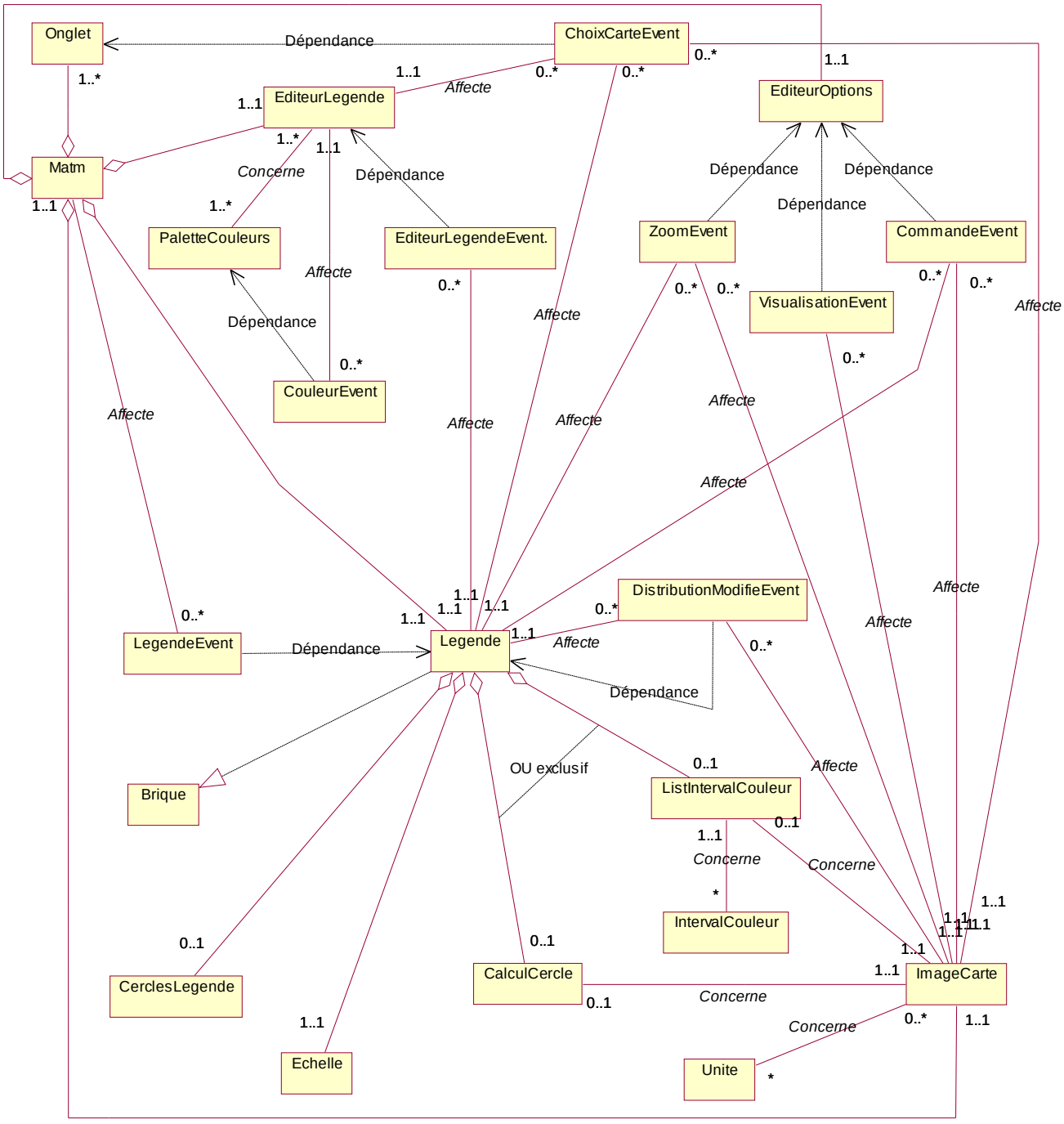


Figure 55 : Diagramme de classes complet sans propriété.

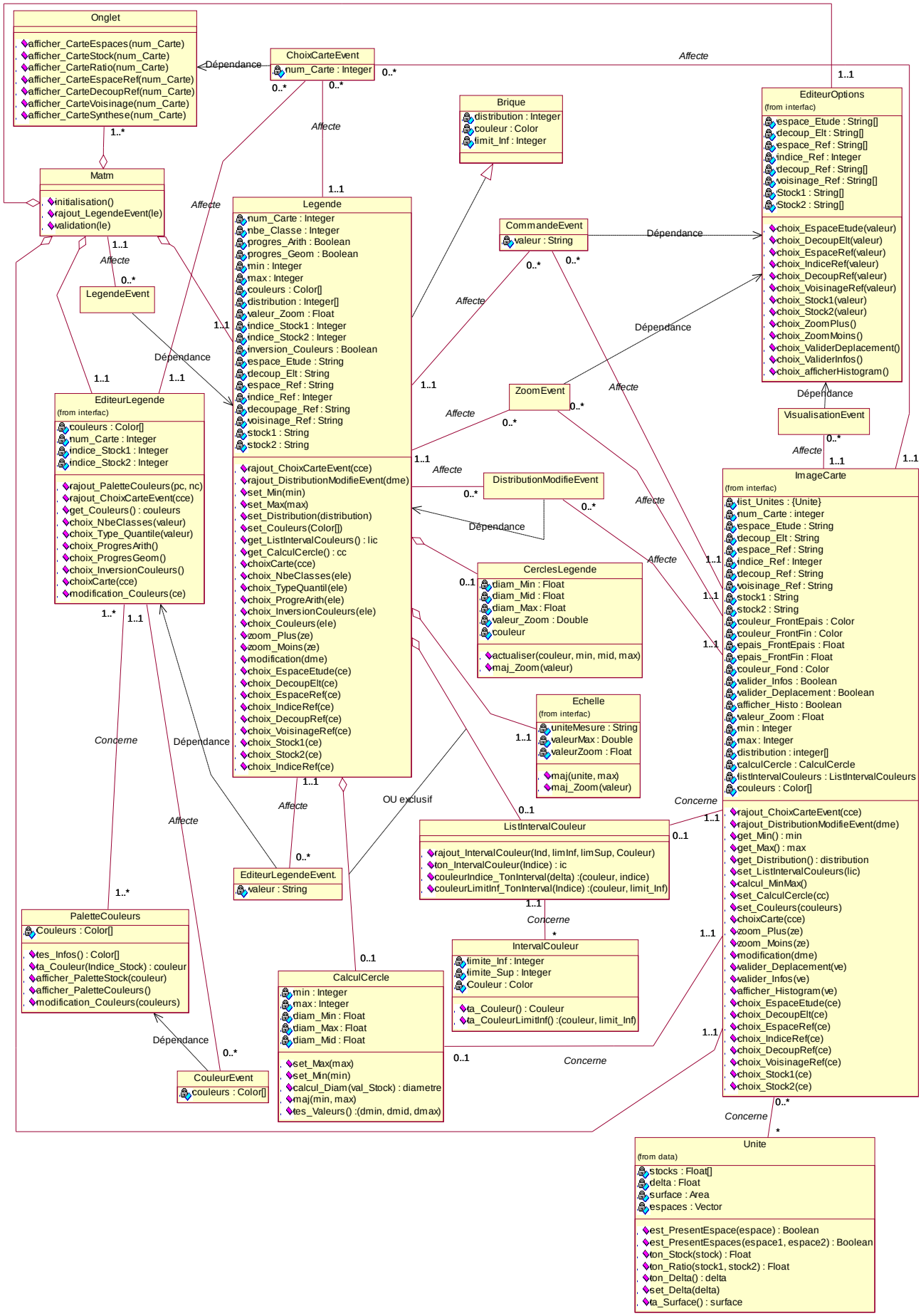


Figure 56 : Diagramme de classes complet avec propriétés.

Description générale :

- Seules les méthodes publiques sont représentées.
- Les constructeurs ne sont pas décrits.
- L'événement *ZoomEvent* dépendant de l'objet de la classe *EditeurOptions* provoque des changements graphiques dans les classes concernées, à savoir *ImageCarte* et *Legende* qui met à jour ses composants *Echelle* et *CerclesLegende* par le biais des méthodes publiques *maj_Zoom(valeur).*
- Les méthodes utilisées pour le cas d'initialisation du système ne sont pas représentées car elles alourdissent le schéma sans apport au niveau de la compréhension générale.
- Le diagramme de classes présenté n'indique pas le cheminement général des évènements entre les différentes classes du système. Dans le cas courant de la modification d'un paramètre dans l'objet de type *EditeurOptions*, l'événement produit (*CommandeEvent*) est écouté par *EditeurLegende* qui met à jour son titre (par exemple). Celui-ci est renvoyé, toujours porteur de l'information sélectionnée par l'utilisateur, à *ImageCarte* qui procède au traitement de son composant. Par la suite, *ImageCarte* créé un événement *ChoixCarteEvent* sans paramètre pour poursuivre le processus de mise à jour de tous les composants du système avant le réaffichage général.

III.2.9 Conclusion

La modélisation d'une interface cartographique reste complexe à réaliser. De plus, nous cherchons à rendre réutilisables des composants par d'autres modules du projet Hypercarte. Les objets sont donc réalisés dans un souci d'autonomie, ce qui induit généralement la duplication d'informations que l'on retrouve dans le diagramme de classe complet. Néanmoins les données partagées sont gérées par le système (e.g. Matm) qui procède à des relations croisées (objets concernés et système).

L'utilisation d'événements nécessite un travail approfondi sur des scénarios possibles de relations entre objets. L'ordre d'arrivée des évènements n'étant pas prévisible nous avons eu recours à des traitements chaînés comme le cas d'utilisation *Choix_CarteEspaces()*. Il simule le traitement des différents composants en séquences ordonnées, ce qui est essentiel pour le déroulement du processus. On essaie ainsi de « greffer » les nouveaux scénarios sur cette séquence (e.g. afficher_CarteEspaceRef) en le complétant au besoin par de nouveaux événements.

Une autre particularité de ce type de développement concerne la création des objets et donc la relation qui existe entre eux. Si pour la modélisation « classique » le processus de création peut (généralement) être facilement décomposé, on créé un professeur puis on lui « attache » des élèves etc. Dans notre interface utilisateur, le moindre zoom fait intervenir une multitude d'interactions entre les objets du système. La décomposition s'en trouve complexifiée, tout comme la modélisation qui en résulte.
Cependant, la phase d'initialisation proposée et les différents cas d'utilisation étudiés répondent bien à leurs objectifs que l'on pourra tester durant la phase d'implémentation et de test (e.g. réalisation) dans le chapitre suivant.

III.3 Réalisation

La réalisation du prototype d'analyse territoriale multiscalaire repose sur les spécifications cartographiques et fonctionnelles présentées en annexe V.2. Nous ne détaillerons donc pas ces contraintes imposées par le cahier des charges, mais rendons compte du résultat obtenu par l'intermédiaire de copie d'écrans en le complétant par divers commentaires. Nous décrivons tout d'abord les principaux composants constituant l'interface, puis les différentes cartes (complètes) obtenues pour finir sur quelques lignes de code représentatives.

III.3.1 Composant EditeurOptions

Le panneau d'options (e.g. classe EditeurOptions) est au cœur de l'application[47], il permet à l'utilisateur de choisir son espace d'étude (Study area), son type de découpage élémentaire (Elementary zoning), son espace de référence ou valeur (Area of reference Or value of reference), le découpage de référence (Zoning of reference), le voisinage de référence (Neighbourhood of reference) ainsi que le ou les indicateurs correspondants aux stocks (Indicator[48]) grâce à des listes de choix (JComboBox). Le panneau d'options est illustré par la figure 57 avec les onglets qui correspondent à l'ensemble des cartes à sélectionner.

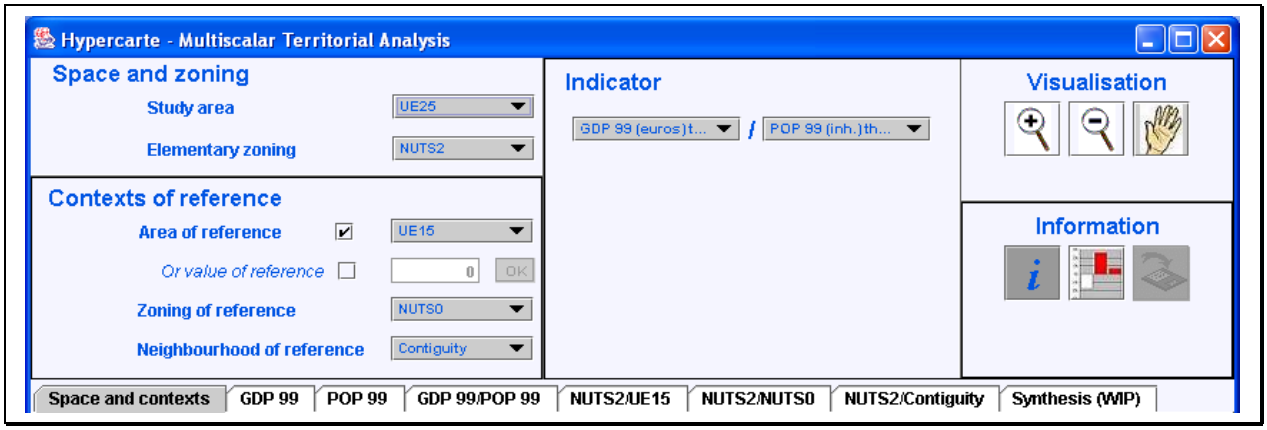


Figure 57 : Composant EditeurOptions avec les onglets.

Par défaut, la carte affichée sera celle de l'espace et des maillages (onglet « Space and contexts »).

Les listes (JcomboBox) sont initialisées au lancement de l'applet à partir des données sérialisées issues du traitement des fichiers fournis par l'équipe du laboratoire Géographie-cités.

Exemple d'incorporation pour la liste des découpages élémentaires ou de références dans la phase de prototypage en vue de test :

- Le vecteur *indicateursDecoupageElement* récupère la liste des *Nuts* disponible grâce à la méthode statique *getListNutsx()* de la classe *AccessData* du package Hypercarte.Data :

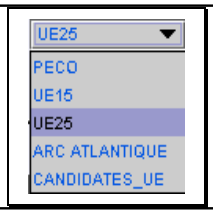
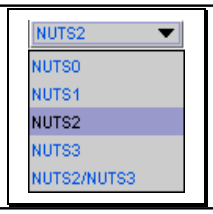
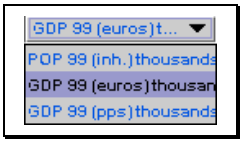
  private Vector indicateursDecoupageElement = AccessData.getListNutsx();

- La liste déroulante est créée par *new JComboBox()* avec le vecteur en paramètre *indicateursDecoupageElement* dans editeurOptions :

  Private JComboBox jComboDecoupageElement =

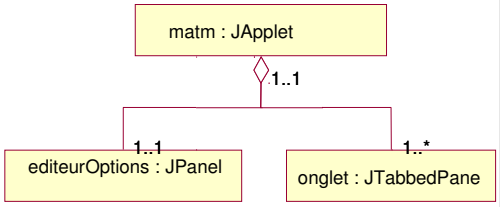
[47] L'applet peut être vu comme une application au sens large
[48] Le premier indicateur correspond au PIB en 1999 (GDP en anglais).

```
new JComboBox(indicateursDecoupageElement);
```

Espace d'étude ou de référence	Découpage élémentaire ou de référence	Indicateur (e.g. stock) 1 ou 2
		

Remarque : le voisinage de référence ne présente à ce jour qu'un seul choix, la contiguïté. Cependant il garde la logique du vecteur (mais ici avec un seul élément).

L'applet du prototype Hypercarte étend le type JApplet et intègre différentes composantes graphiques comme le montre la figure ci-dessous.

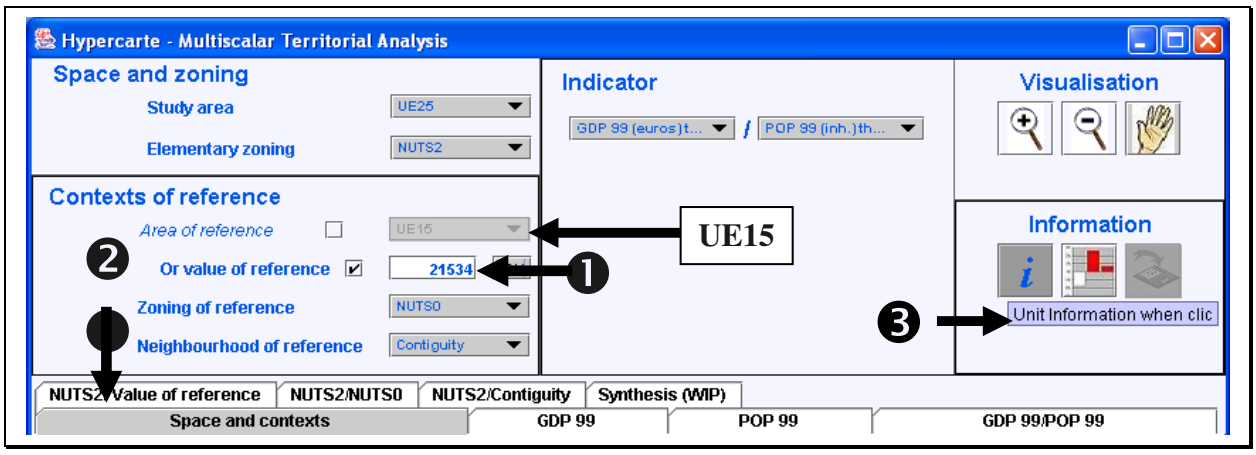
Composantes graphiques	Codage simplifié correspondant
	(1) public class Matm extends JApplet implements … (2) private EditeurOptions editeurOptions = new EditeurOptions(); (3) private JTabbedPane onglet = new JTabbedPane(); (4) contentPane.setLayout(new BorderLayout()); (5) contentPane.add(editeurOptions, BorderLayout.NORTH); (6) contentPane.add(onglet);

Explication :

(1) la classe Matm dérive de JApplet et implémente ses différents listeners
(2) création de l'objet editeurOptions de type JPanel
(3) création de l'objet onglet de type JTabbedPane
(4) la politique de positionnement choisie est de type BorderLayout
(5) l'objet editeurOptions est ajouté dans la fenêtre principale au nord
(6) l'objet onglet[49] est ajouté dans l'espace restant (c'est-à-dire tout le reste).

Pour la représentation suivante, seul l'espace de référence a été remplacé pour choisir une valeur de référence (21534) ❶. Celle-ci correspondra à la nouvelle valeur de référence (indice 100) pour la déviation notée précédemment NUTS2/UE15 (onglet) et maintenant NUTS2/Value of reference ❷.
Remarque : la chaîne de caractères affichée étant plus grande, les onglets se positionnent suivant 2 lignes, en gérant l'espace suivant la politique des conteneurs.

[49] L'onglet est vu au sens large, il contient la carte et ses légendes.



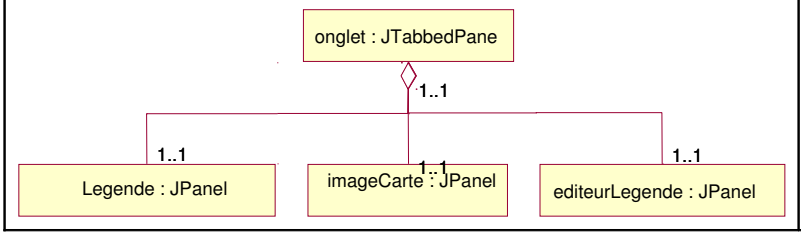
Remarques :

- en positionnant la souris un certain temps sur le bouton "i" comme information, une aide en ligne apparaît sous forme de bulle (Unit information when clic) ❸.
- chaque bouton possède 2 représentations (e.g. image) suivant sa sélection ou non.
- une sélection dans la liste des espaces de référence (e.g. UE15) calcule automatiquement la valeur de l'indice de référence et l'intègre dans la zone "Or value of reference" à titre indicatif. Dans le cas présent, le ratio calculé du PIB/nombre d'habitants pour l'Europe des 15 est de 21534 ❶. Il représentera l'indice 100 par la suite.

L'editeurOptions n'étant qu'un objet de type interface utilisateur, toute nouvelle sélection dans une liste sera enregistrée dans le champ correspondant des classes ImageCarte et Legende qui « l'écoutent ». Il en va de même pour les boutons de déplacement, d'information et d'histogramme (en valeur booléenne) mais uniquement pour la classe ImageCarte.

III.3.2 Composant EditeurLegende

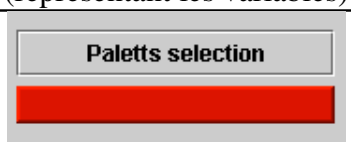
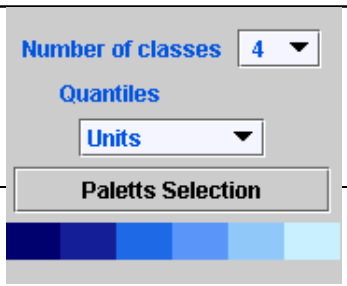
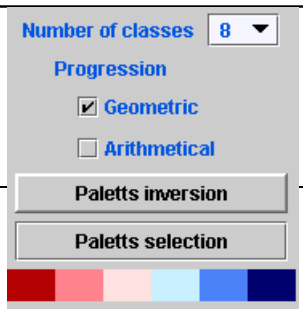
Comme le composant EditeurOptions, la classe EditeurLegende offre des services à l'utilisateur via son interface et ses choix. Elle est de type JPanel comme le montre la figure ci-dessous.

Composantes graphiques	Commentaire
	L'objet onglet est composé : - d'une légende - d'une carte - d'un éditeur de légende on garde comme politique de positionnement : BorderLayout.

Les différentes représentations de l'interface graphique de la classe EditeurLegende, suivant les familles de carte désirées, sont présentées ci-dessous selon 2 états.

- A l'état initial, la ou les couleurs en cours sont affichées (figure 58).

Figure 58 : Composant EditeurLegende à l'état initial.

Carte des stocks (représentant les variables)	Carte de ratio	Carte des déviations
		

Une carte de stock, représentée par des cercles, ne nécessite qu'une couleur contrairement aux autres cartes qui traitent les informations par tranches de valeurs associées chacune à une couleur différente.

- Quand on sélectionne le bouton choix de palette de couleurs (Paletts selection), l'objet de type PaletteCouleurs change d'état et attend une sélection correspondant à une (carte des stocks) ou à plusieurs couleurs parmi des gammes proposées (figure 59).

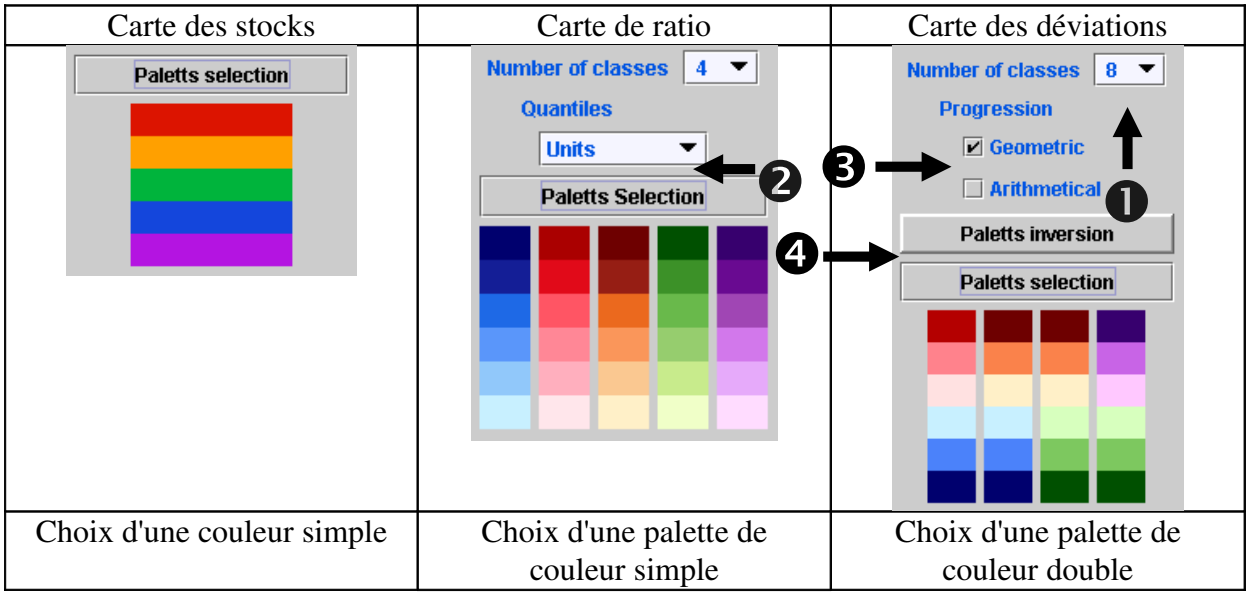


Figure 59 : Composant EditeurLegende à l'état « choix de couleurs ».

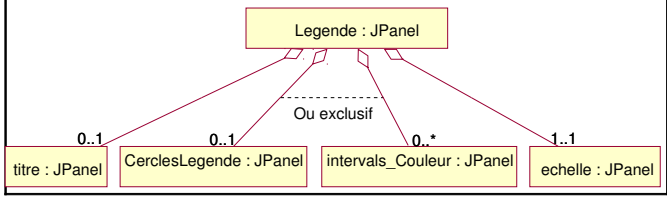
- Lorsque l'utilisateur sélectionne une couleur ou une palette de couleur, l'objet de type PaletteCouleurs utilise cette valeur par défaut et l'affiche suivant la configuration décrite au premier affichage (e.g. état initial au repos). Cette action produit un événement CouleurEvent() qui sera « écouté » par l'*EditeurLegende*.

Remarque : il n'existe pas de choix de couleur pour la carte de l'espace et des maillages qui sera toujours affichée en noir et gris. La carte de synthèse est représentée en couleur mais n'offre pas la possibilité de changer de gamme (choix délibéré).

Les deux listes de choix (JComboBox) pour le nombre de classes (de 2 à 10) (figure 59) ❶ et le quantile choisi (Unités, stock1 ou stock2) ❷ ainsi que le type de progression choisi (arithmétique ou géométrique) ❸ et l'inversion des couleurs ❹ génèrent des évènements qui seront écoutés par la classe Legende qui mettra à jour les champs concernés en vue des futurs calculs.

III.3.3 Composant Legende

La classe Legende représente, de manière graphique (hormis les titres), la résultante des choix exprimés par l'utilisateur via les classes EditeurOptions et EditeurLegende. Il est de type JPanel comme le montre la figure ci-dessous.

Composantes graphiques	Commentaire
	L'objet Legende est composé : - d'une zone de titre - soit d'un « CerclesLegende » dans le cas de la représentation de stock - soit d'un ensemble d'intervalsCouleur pour les autres cartes (représentées en aplats de couleur).

	- d'une échelle qui est toujours représentée.

La représentation des différentes possibilités offertes est affichée en fonction des cartes rencontrées (figure 60).

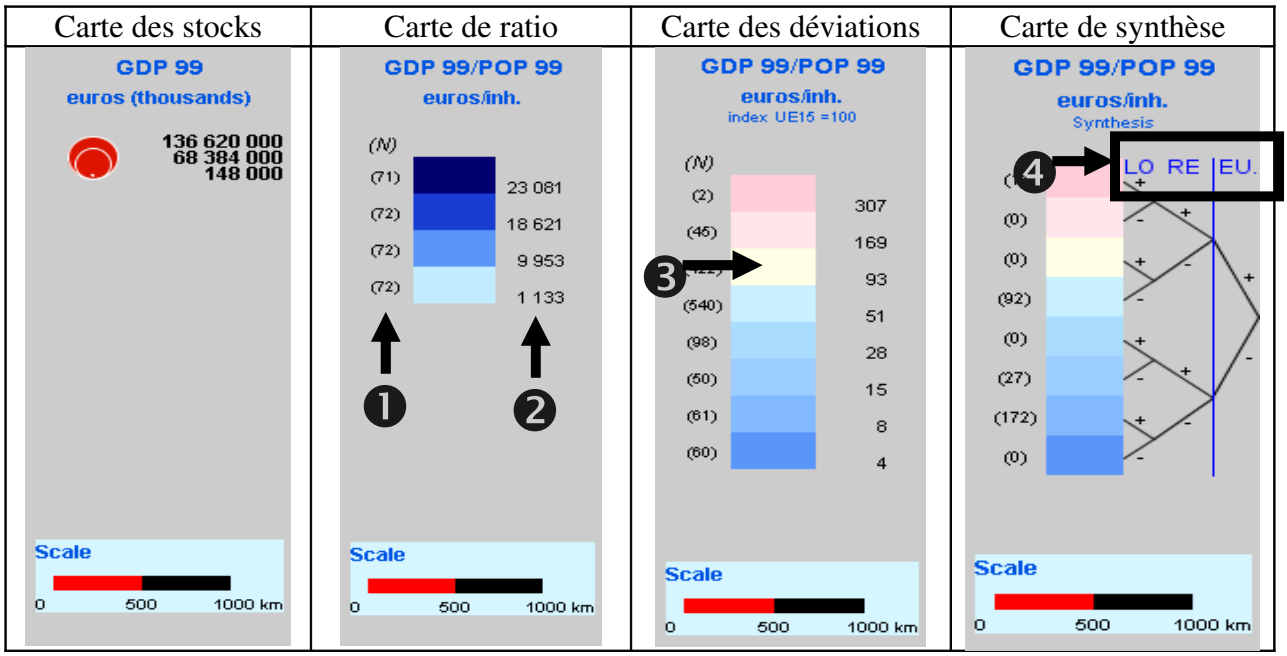


Figure 60 : Composant Legende.

Pour les aplats de couleurs (e.g. les briques), on retrouve le nombre d'individus (la fréquence) appartenant à l'intervalle concerné à gauche ❶, la couleur et la limite inférieure de l'intervalle considéré ❷. L'intervalle qui englobe la valeur 100 (indice de référence) est représenté par une couleur neutre (ici, le jaune pour les cartes de déviations et contiguïté) ❸.

La *Legende* de la synthèse se lit de la manière suivante :

- l'unité (exemple, le département) est testée par rapport à la moyenne des unités qui lui sont contiguës (déviation **LO**cale ❹) ce qui donne comme résultat le booléen + ou -,
- cette même unité est comparée à son unité supérieure (ici **RE**gional ❹) et produit un résultat booléen + ou -,
- cette même unité est comparée à l'espace **EU**ropéen ❹ qui entraîne le résultat booléen + ou -.

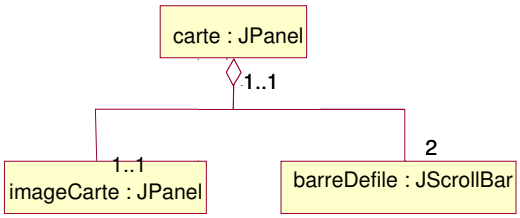
Ces différentes déviations sont représentées par un graphe binaire et dépendent des paramètres sélectionnés.

Remarque : cette classe Legende est la seule à ne permettre aucune interaction avec l'utilisateur.

III.3.4 Cartes affichées

Les cartes (e.g. classe ImageCarte), comme les autres composants graphiques déjà observés, héritent de la classe Jpanel de Swing.

Composantes graphiques	Commentaire

	L'objet carte est composé : - d'une imageCarte (la représentation graphique de la carte) - et de 2 barres de défilement, horizontale et verticale (JScrollBar) *Remarque* : l'objet imageCarte est lui-même composé de 2 décorations bleues souhaitées à gauche et à droite (non représentées)

III.3.4.1 Carte de l'espace et des maillages

En fonction de l'espace d'étude observé, ici l'Europe des 25, on présente (figure 61) quelques exemples d'affichage en jouant sur les paramètres découpage élémentaire et découpage de référence.

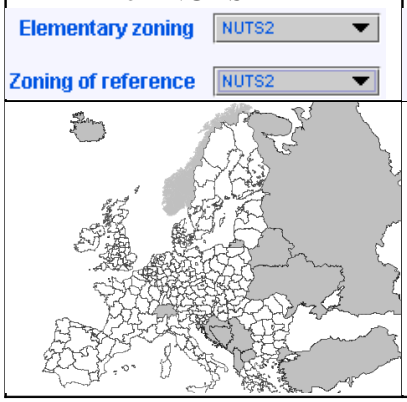
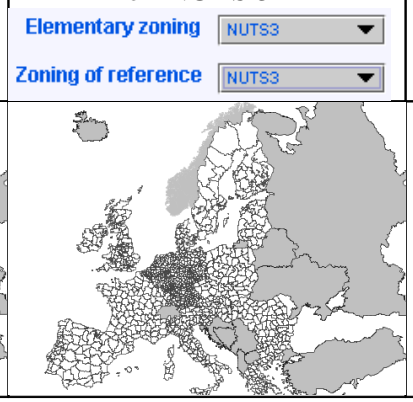
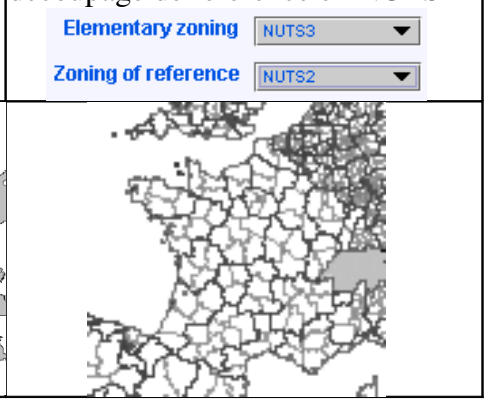
Affichage des découpages (élémentaire et de référence), en NUTS 2	Affichage des découpages (élémentaire et de référence), en NUTS 3	Affichage du découpage élémentaire en NUTS 3 et du découpage de référence en NUTS 2
		

Figure 61 : Carte de l'espace et des maillages.

Remarque : l'affichage agrandi (à l'extrême droite) permet de distinguer les valeurs différentes utilisées pour le tracé du contour des unités. Les frontières des départements sont affichées avec un contour gris fin et les régions avec un contour plus épais et noir.

III.3.4.2 Carte des stocks

La représentation des variables (e.g. Indicator) se fait à l'aide de cercles proportionnels par leur surface, à la valeur du stock représenté. La figure 62 illustre le nombre d'habitants pour l'année 1999 exprimé en millier ❶.

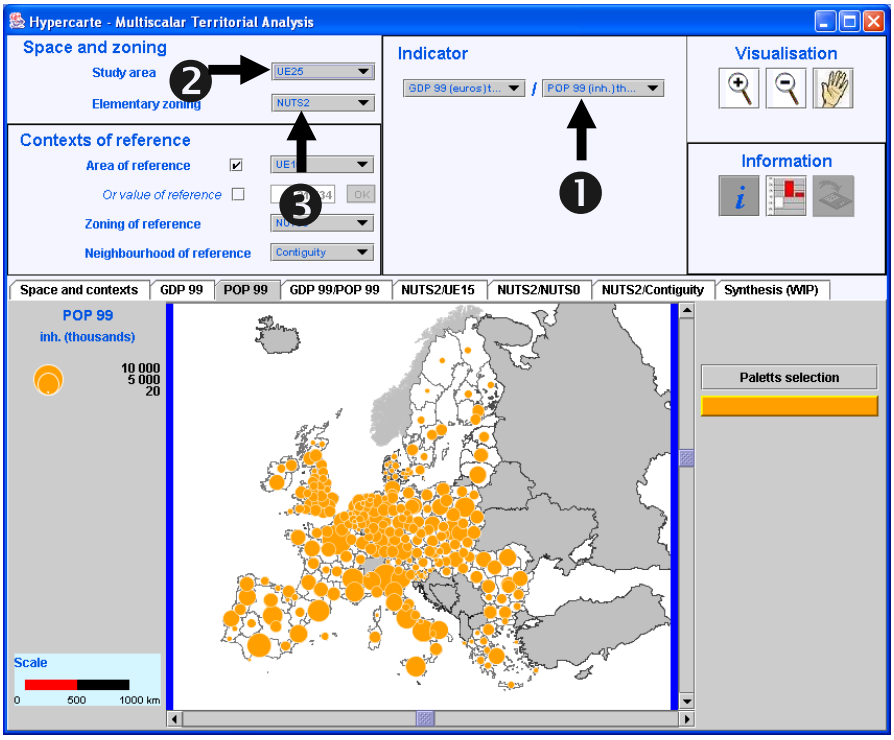
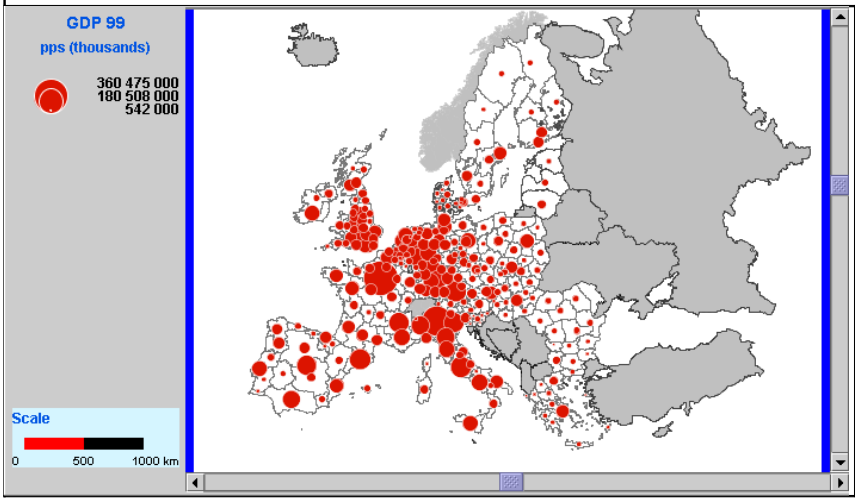


Figure 62 : Carte des stocks.

L'espace étudié sur lequel porte l'analyse concerne l'Europe des 25 (UE25 ❷) représenté en fond blanc. Les unités territoriales sont affichées suivant le découpage élémentaire sélectionné (ici en NUTS2 ❸ pour les régions).

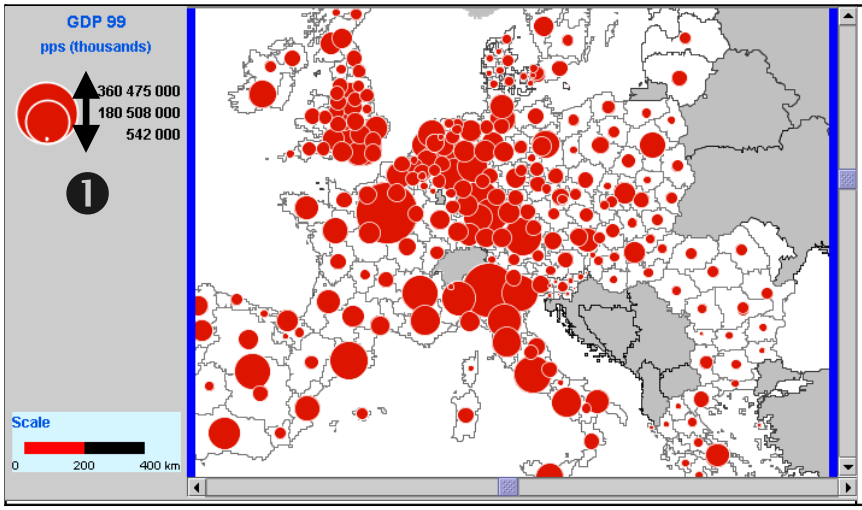
En fonction de la valeur de zoom utilisée, l'affichage du « tandem » carte et légende possède des particularités souhaitées par les décideurs. Elles sont présentées sous la forme des figures ci-dessous et commentées.

La valeur du zoom est à l'état initial



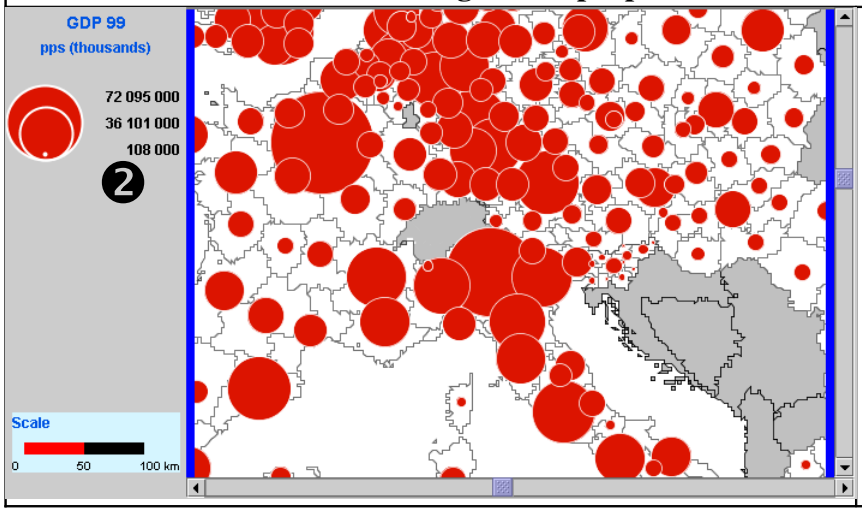
- les tailles (max et min) des cercles de la carte correspondent aux tailles (max et min) des cercles de la légende.
- les valeurs (max et min) de la légende correspondent aux valeurs (max et min) des unités représentées.
- l'échelle (scale) est à l'état initial.
- les barres de défilement sont à l'état initial.

La valeur du zoom est inférieure à une limite fixée : les cercles de la légende et de la carte sont augmentés proportionnellement



- les tailles (max et min) des cercles de la carte correspondent aux tailles (max et min) des cercles de la légende.
- les valeurs (max et min) de la légende correspondent aux valeurs (max et min) des unités représentées.
- l'échelle est mise à jour.
- les barres de défilement sont mises à jour[50].
- l'espacement des données affichées de la légende est modifié en hauteur ❶.

La valeur du zoom est supérieure à la limite fixée, seuls les cercles de la carte sont augmentés proportionnellement



- les tailles des cercles de la carte ne correspondent plus aux tailles des cercles de la légende qui ont atteint le diamètre maximum autorisé.
- les valeurs de la légende décroissent proportionnellement en fonction du zoom ❷.
- l'échelle est mise à jour.
- les barres de défilement sont mises à jour.
- l'espacement des données affichées de la légende est modifié en hauteur.

Remarque : les cercles les plus gros sont affichés les premiers (besoin de lisibilité). Le traitement relatif à l'affichage des cercles est donc ordonné suivant la valeur statistique de l'unité (d'où un traitement de tri intermédiaire).

III.3.4.3 Carte de ratio

Le ratio de deux variables est représenté par l'intermédiaire de quantiles. La figure 63 montre le ratio (PIB/ Nombre d'habitant) visualisé à l'aide d'une gamme de couleurs simple (ici un dégradé de bleu) suivant 4 classes.

[50] Le taquet de sélection de la barre de défilement est réduit proportionnellement en fonction du zoom.

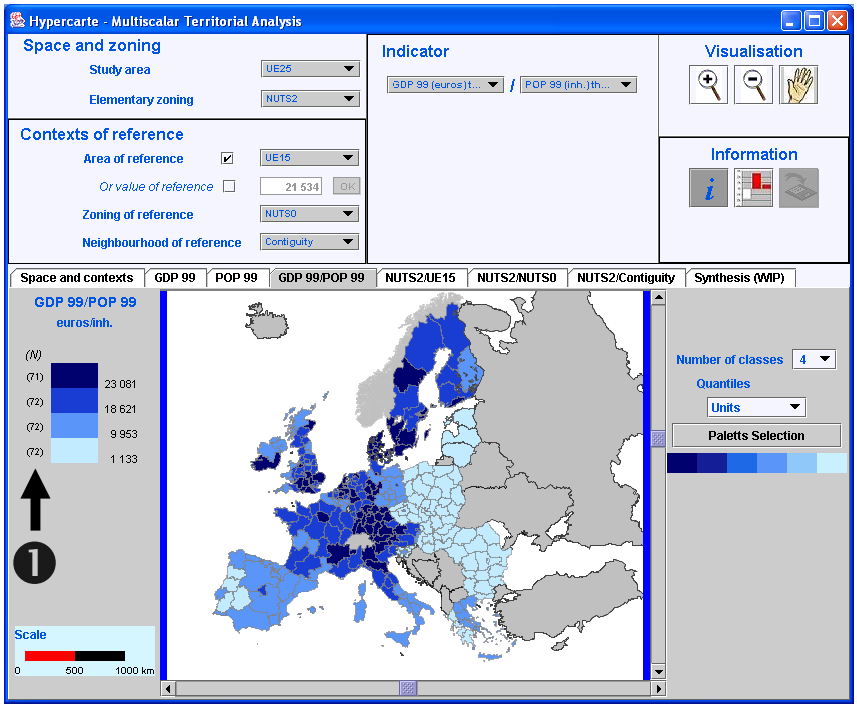
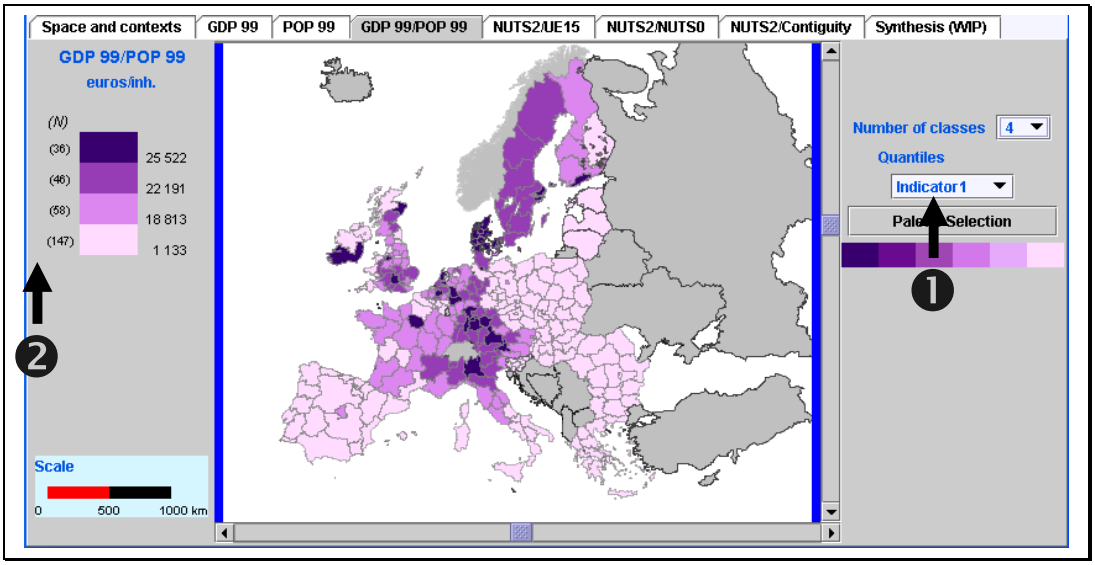


Figure 63 : Carte de ratio.

Remarque : les effectifs par tranche de classes doivent être les mêmes (à 1 près dans le cas présenté : 3 tranches à 72, et 1 à 71 ❶). La définition du quantile est prise dans son sens courant (classe de même effectif).

Dans la représentation suivante, seuls le choix de la gamme de couleurs et le type de quantile ont changé. Ici *Indicator1* ❶ correspond au premier indicateur, c'est-à-dire le PIB en 1999. Chaque classe (au nombre de 4) devra donc rassembler un quart de la richesse totale (PIB) mais ce sera toujours le ratio PIB/habitant qui sera représenté, d'où des effectifs de classes différents ❷.



III.3.4.4 Carte de déviations hiérarchiques

L'analyse des déviations hiérarchiques est présentée suivant le découpage élémentaire des régions (NUTS2) ❶ et l'espace de référence (EU15) ❷. La représentation obtenue (figure 64) est définie avec 8 classes ❸ et une progression de type géométrique ❹ suivant une gamme de couleur double (rouge – bleue) ❺.

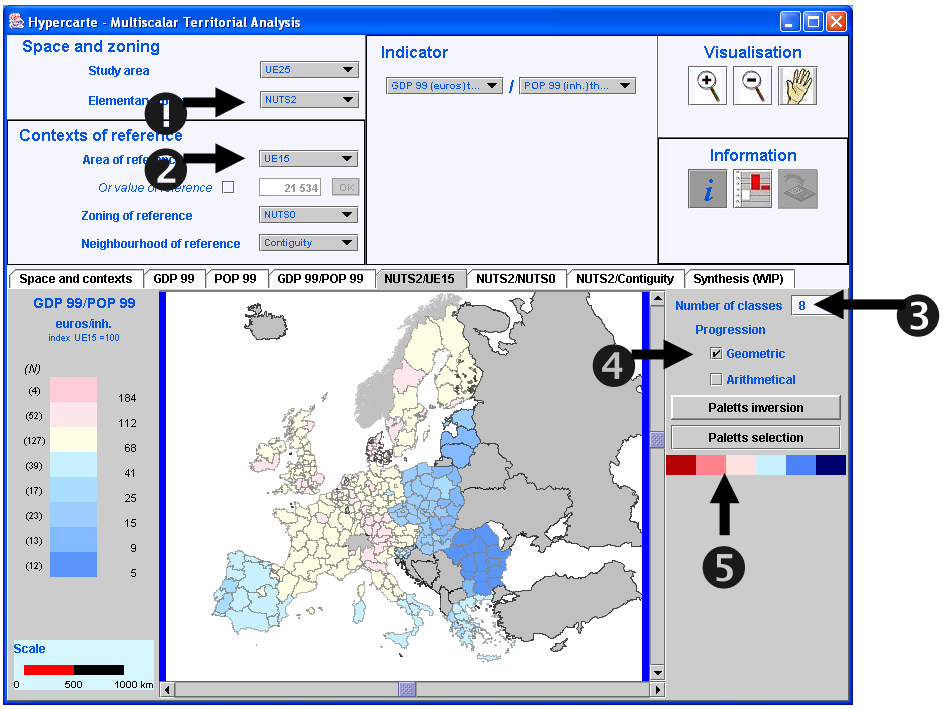


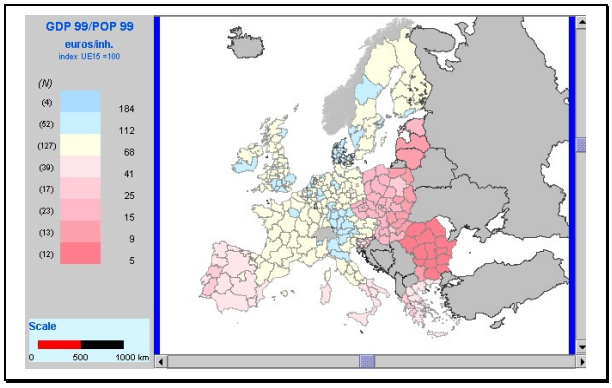
Figure 64 : Carte de déviation hiérarchique au niveau global.

La formule mathématique appliquée pour la déviation au niveau global (ici, l'Europe des 15) est [ATM03] :

$$I_i^{Eur} = 100 \times \frac{\frac{PIBi}{POPi}}{\frac{\sum_j Eur_j \times PIB_j}{\sum_j Eur_j \times POP_j}} \quad (1)$$

Explication : la déviation d'une région donnée (i) au niveau européen est définie par l'équation (1). Dans le cas présent, le secteur de référence est égal à EU15, la variable booléenne Eur_j prend la valeur 1 pour les régions appartenant à EU15 et la valeur 0 pour les régions des autres pays. Il est ainsi possible de produire d'autres cartes avec EU25 ou EU27 comme nouvelle référence.

Ci-dessous, la carte présentant les mêmes caractéristiques mais en inversant les couleurs (*Paletts inversion*).



La figure 65 présente une analyse des déviations hiérarchiques suivant le découpage élémentaire des régions (NUTS2) mais en le comparant avec le découpage de référence NUTS0, c'est-à-dire le pays d'appartenance de l'unité. La représentation est définie avec 4 classes et une progression de type arithmétique présentée avec une gamme de couleur double (rouge – bleue).

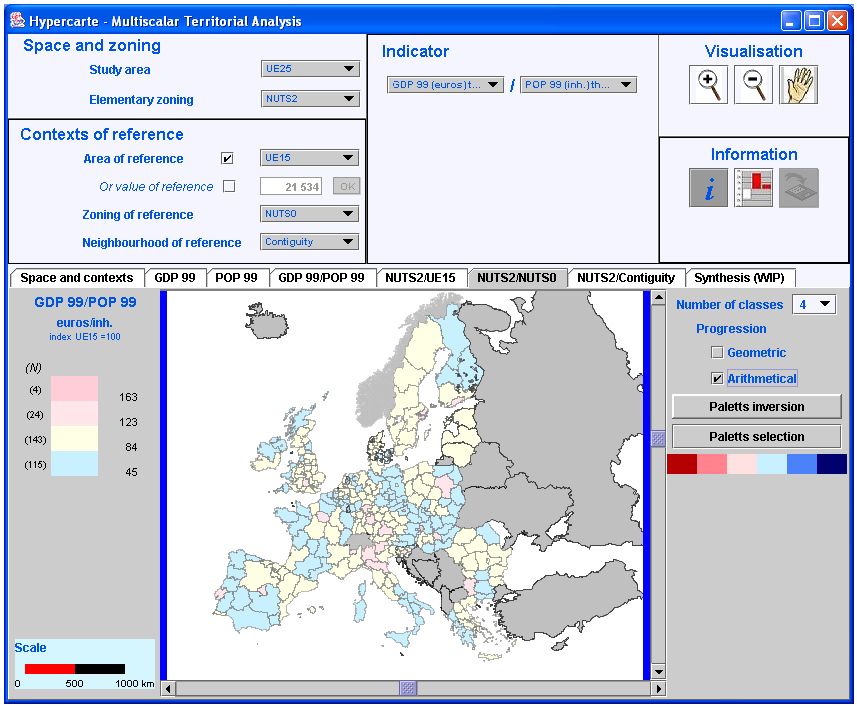


Figure 65 : Carte de déviation hiérarchique au niveau intermédiaire.

La formule mathématique appliquée pour la déviation au niveau intermédiaire (ici, la nation) est [ATM03] :

$$I_i^{Nat} = 100 \times \frac{\frac{PIBi}{POPi}}{\frac{\sum_j Nat_{ij} \times PIB_j}{\sum_j Nat_{ij} \times POP_j}} \quad (2)$$

Explication : la déviation d'une région donnée (i) au niveau national est définie par l'équation (2). La variable booléenne Nat_{ij} décrit si deux espaces i et j appartiennent au même secteur hiérarchique. Dans notre cas, elle est définie au niveau de l'état (e.g. le Pays), ainsi, Nat_{ij} prend la valeur 1 quand sont traités une région et son pays d'appartenance et la valeur 0 dans les autres cas.

Remarque : le nombre de représentations de cartes de déviations hiérarchiques n'est limité que par le nombre d'indicateurs présents dans le système. Il n'y a aucune limite fonctionnelle : toutes les combinaisons sont donc permises.

III.3.4.5 Carte de déviation locale

Cette famille de cartes représente la déviation locale (unités contiguës). La figure 66 illustre cette analyse pour les départements (NUTS3) de l'Europe des 15 (espace d'étude).

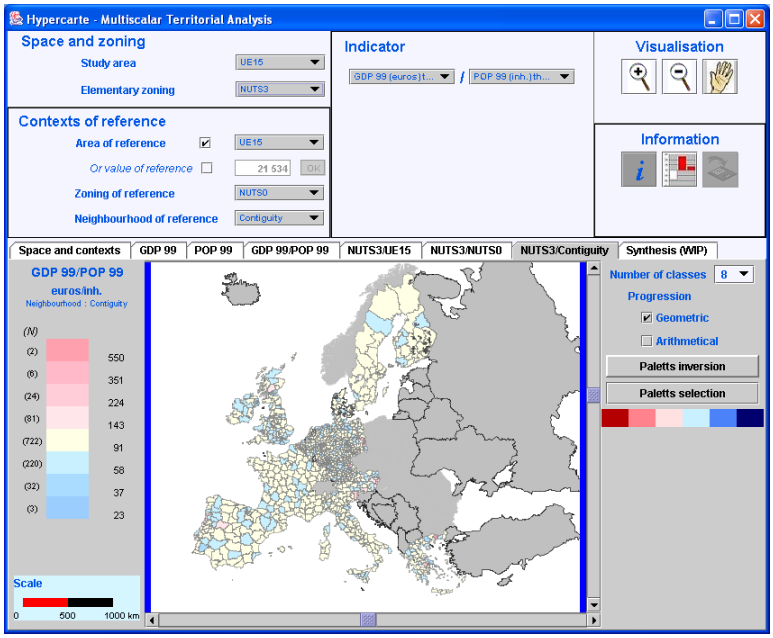


Figure 66 : Carte de déviation au niveau local.

La formule mathématique appliquée pour la déviation au niveau local (frontière commune) est [ATM03] :

$$I_i^{Loc} = 100 \times \frac{\frac{PIBi}{POPi}}{\frac{\sum_j Loc_{ij} \times PIB_j}{\sum_j Loc_{ij} \times POP_j}} \quad (3)$$

Explication : la déviation d'une région donnée (i) au niveau local est caractérisée par l'équation (3). La variable booléenne Loc_{ij} définit le *niveau potentiel d'interactions locales* entre *i* et *j*. Dans notre cas, ce niveau local d'interaction correspond à la valeur 1 pour les régions contiguës (e.g. voisines) et la valeur 0 pour les autres.

III.3.4.6 Carte de synthèse

Cette famille de carte représente la synthèse des déviations obtenues précédemment (figure 67).

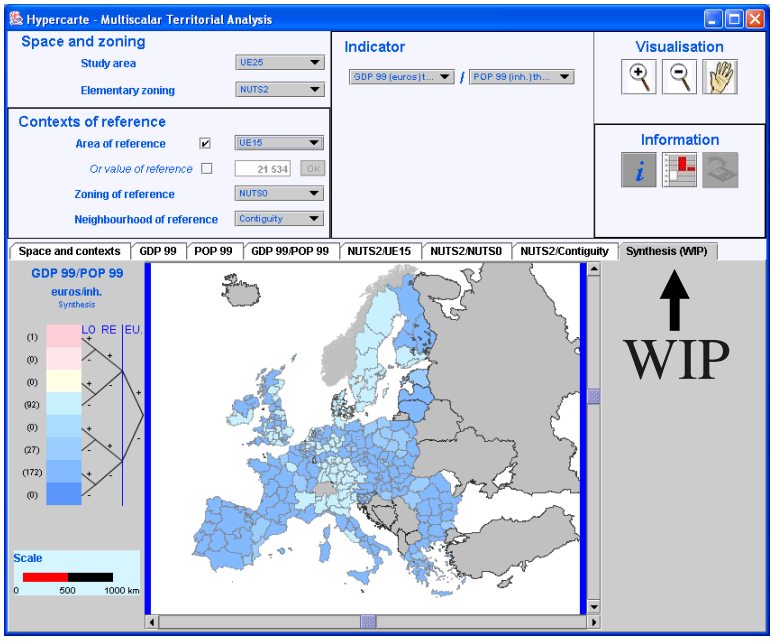
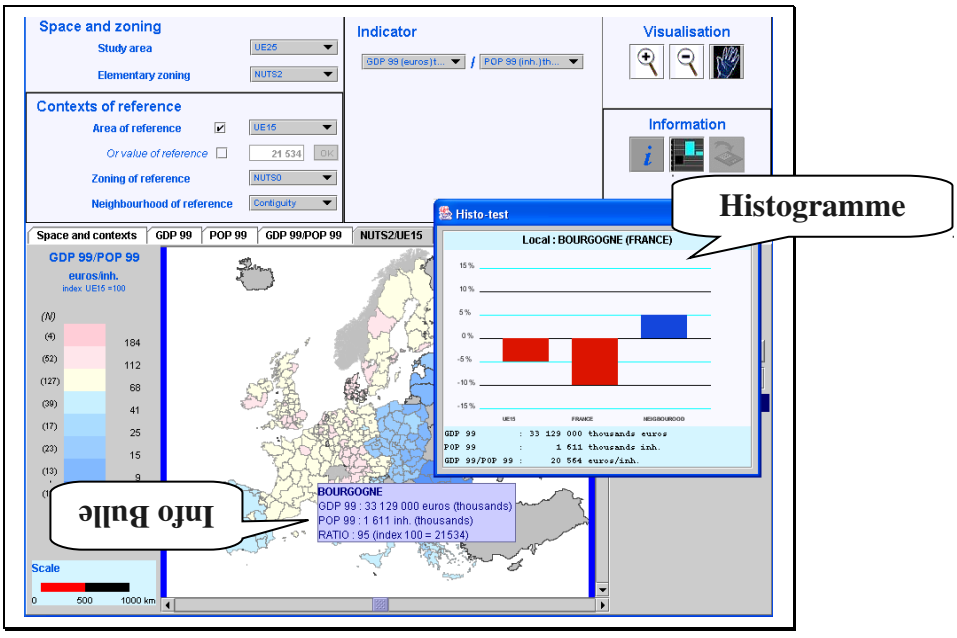


Figure 67 : Carte de synthèse.

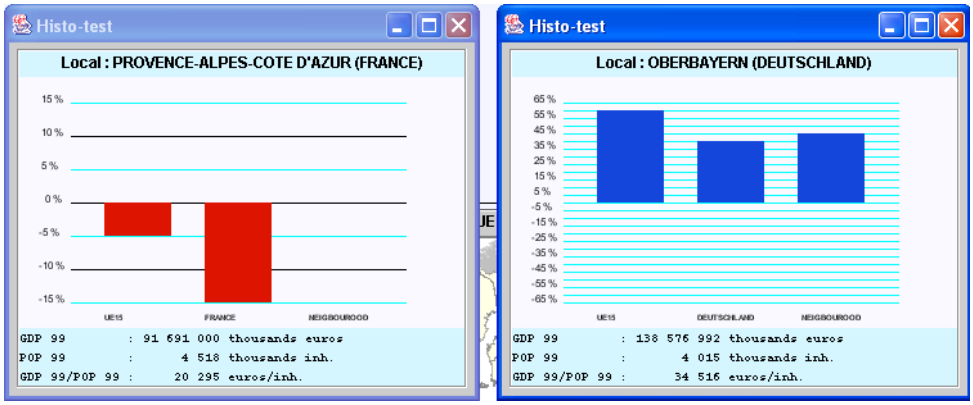
Remarque : sur l'onglet on peut apercevoir "Synthesis **W**ork **I**n **P**rogress" car cette carte est la seule à n'avoir pas été validée par les géographes.

III.3.4.7 Interactions utilisateurs sur la carte

L'utilisateur peut déplacer la carte avec sa souris par « cliqué-déplacé » et obtenir des informations sur l'unité sélectionnée grâce à un histogramme (annexe V3) et/ou à une « info bulle ».



- Les informations localisées lors d'une sélection sur des unités sont représentées par les histogrammes ci-dessous.



Il existe autant d'histogrammes que désirés (nouvelle fenêtre). L'histogramme représente les mêmes informations quelque soit la carte sélectionnée. L'échelle en ordonnée est variable selon le maximum et le minimum à représenter (voir ci-dessus les lignes horizontales)

- Une information localisée lors d'un « survol » de souris sur une unité est représentée par une « info bulle ». Les informations de la bulle sont, elles, contextuelles suivant la carte en cours. Le tableau ci-dessous décrit pour la même unité survolée (ici, Ile de France) les informations affichées suivant la carte sélectionnée.

Carte de l'espace et des maillages	**ILE DE FRANCE** GDP 99 : 382 122 016 euros (thousands) POP 99 : 10 964 inh. (thousands)
Carte du stock GDP 99 (e.g. PIB 99)	**ILE DE FRANCE** GDP 99 : 382 122 016 euros (thousands)
Carte du stock POP 99	**ILE DE FRANCE** POP 99 : 10 964 inh. (thousands)
Carte de ratio	**ILE DE FRANCE** GDP 99 : 382 122 016 euros (thousands) POP 99 : 10 964 inh. (thousands) GDP 99/POP 99 : 34 852 euros/inh.
Carte de déviation hiérarchique (par rapport à l'espace de référence : Europe des 15)	**ILE DE FRANCE** GDP 99 : 382 122 016 euros (thousands) POP 99 : 10 964 inh. (thousands) RATIO : 161 (index 100 = 21534)
Carte de déviation hiérarchique (par rapport au découpage de référence : NUTS0)	**ILE DE FRANCE** GDP 99 : 382 122 016 euros (thousands) POP 99 : 10 964 inh. (thousands) RATIO : 146 (index 100 = NUTS0)
Carte de déviation locale	**ILE DE FRANCE** GDP 99 : 382 122 016 euros (thousands) POP 99 : 10 964 inh. (thousands) RATIO : 100
Carte de synthèse	Non réalisée

III.3.4.8 Exemples de codes liés à l'affichage

Les exemples donnés à titre indicatif correspondent à un affichage standard pour la première méthode. La seconde décrit comment on peut améliorer le rendu et comment utiliser un zoom ou une translation.

Un affichage des contours avec une épaisseur et une couleur donnée

```
/** Affiche une carte en fonction d'un niveau de nuts et d'une liste d'index
 * @param nutx : nom du niveau de nutx pour le tracer
 * @param listIndex : liste des index a tracer
 * @param g : espace de dessin
 * @param color : type de la couleur du tracé
 * @param epais : valeur pour l'épaisseur du tracé du contour

 */
private void buildImage(String nutx, Vector listIndex, Graphics2D g, Color color, Float epais){
      Vector v = getArea(nutx, listIndex); // une aire (Area) est réalisée à la volée à partir de points
      if (v != null){
          g.setColor(color);
          BasicStroke stroke = new BasicStroke(epais);   // Constructeur pour parametrer l'épaisseur.
          for (Enumeration e = v.elements(); e.hasMoreElements();){ // Parcours du vecteur
              Area frontiere = (Area) e.nextElement();   // Area représente une surface
              g.draw(frontiere);   // on dessine le contour en couleur
          }
      }
      else{
          // traitement de l'exception...
      }
   }
```

Fonction d'affichage graphique avec gestion du zoom et des translations

```
 public void paint(Graphics g2)   // ou paintComponent …
{
  super.paint(g2);
  Graphics2D g = (Graphics2D)g2;   // Conversion de Graphics en Graphics2D
  g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,    // On utilise l'antialiasing (exemple)
  RenderingHints.VALUE_ANTIALIAS_ON);                    // pour augmenter la précision du dessin.

  g.translate(this.getTranslateX(), this.getTranslateY()); // Translation paramétrée
  g.scale(this.getScale(), this.getScale());                    // Zoom paramétré

  BasicStroke stroke = new BasicStroke((float)this.getThick()); // Epaisseur paramétrée
  g.setStroke(stroke);
(…)
```

Une autre manière possible de gérer un zoom est présentée avec la classe AffineTransform :

AffineTransform t = AffineTransform.getScaleInstance(scaleValue, scaleValue);
Area a = unite.getSurface();
a.transform(t);

Avec : Value représente la valeur du zoom à effectuer.

III.3.5 Conclusion

La réalisation des différentes fonctionnalités de l'interface cartographique présentée a été validée tout au long du mémoire par les différentes équipes concernées. La conception des différentes classes ne pose pas de problème particulier. Cependant, la gestion des différents évènements nécessite une réflexion poussée. Elle a été établie lors des phases de modélisation et entraîne notamment une bonne utilisation des commandes de rafraîchissement (Paint ou update) en fin de transaction. En terme de performance, le premier affichage d'une carte prend de 1 à 3 secondes pour une machine de bureau standard en exploitant le fichier sérialisé le plus défavorable de 15 Mo.

IV CONCLUSION ET PERSPECTIVES

Le module d'analyse territoriale multiscalaire présenté dans ce document se propose de valider la dimension opérationnelle d'analyses dites multiscalaires par le biais de données statistiques socio-économiques. Si cet objectif s'avère majeur d'un point de vue politique, produire de manière automatique et interactive un ensemble de familles de cartes proposant des déviations entre plusieurs niveaux d'échelles relève plus de la compétence des « informaticiens ».

La conception et réalisation d'un tel projet mettent en évidence plusieurs difficultés et points à souligner.
L'affichage de données cartographiques mono couche ne pose pas réellement de problème avec un langage de développement comme Java dans la mesure où l'on comprend bien comment sont traités les objets. La représentation multi échelles et les traitements multiscalaires imposent, eux, une réflexion plus poussée.
La représentation multi échelles nécessite de différencier, pour l'utilisateur, les différentes couches en présence par le biais de couleurs et d'épaisseurs appliquées aux frontières respectives. Pour le système, ces différents maillages sont représentés suivant un arbre sur lequel se portent les différentes déviations relatives à l'analyse hiérarchique.
Les mécanismes mis en jeu pour l'analyse spatiale se réfèrent aux seules données géographiques, imposant des notions de topologie qui peuvent être rendues soit explicitement dans le cas d'un SGBD, soit implicitement comme la solution retenue (e.g. calculées à la volée).
L'architecture du système client serveur pose la problématique du « qui fait quoi ? » afin notamment d'optimiser les procédures de calcul et l'échange de données entre machines distantes. La réalisation effectuée à partir d'un jeu de test grandeur nature a pu valider le passage à l'échelle dans les conditions les plus défavorables pour l'utilisateur final (e.g. toutes les données sont transmises).
La dernière difficulté consistait à établir une interface cartographique interactive permettant d'obtenir une carte ergonomique et explicite pour l'utilisateur final. Celle-ci a été validée tout au long du développement par les compétences des cartographes du laboratoire Géographie-cités.

Même si l'application produit les résultats escomptés, des perspectives d'amélioration, d'évolution, et de réflexion sont identifiables.
L'utilisation d'un SGBD au niveau du serveur de données permettrait une mise à jour des données plus standardisée.
Une structuration des données semblable à l'approche décrite dans la section II.1.5.2.2.3 apporterait une topologie rendue explicitement capable d'exprimer d'autres formes d'analyse spatiale que la seule contiguïté.
Au lancement de l'application, le serveur fournit l'applet ainsi que l'ensemble des données nécessaires à la constitution de cartes. Si ce choix permet aisément de diffuser le produit, sur Cédérom par exemple (spécificité demandée), une interaction plus fine entre le client et le serveur via un SGBD (e.g. ne remonter qu'un sous ensemble de données en fonction des choix utilisateurs) semblerait intéressant à tester du point de vue performance.
Une réflexion intéressante concerne la représentation de la donnée cartographique pour le projet Hypercarte. Le mode vectoriel est particulièrement adapté à des zones bien définies comme c'est le cas pour le module ATM, mais il en va tout autrement pour représenter les phénomènes continus du module d'analyse spatiale multiscalaire. Même si l'utilisation d'un langage comme Java permet ces réalisations, il est cependant nécessaire d'observer les avancées du standard SVG. Cette alternative soutenue par le W3C reste possible pour ces deux modules et allègerait les principaux traitements de visualisation (zoom, déplacement …).

A l'issue de ce travail, on peut considérer que l'objectif de fournir un applicatif sous logiciel libre et multi plates-formes permettant de démocratiser un accès à l'information géographique est

atteint. Cependant, les nombreuses améliorations identifiables feront l'objet d'une prochaine étude réalisé par un stagiaire ingénieur CNAM, au sein de l'équipe LSR, pour concourir à un produit plus finalisé.

V ANNEXES

V.1 Représentation cartographique suivant deux modes d'analyse

La figure V.1 représente, sous la forme de cartes, le rapport du Produit National Brut par habitant en Italie en 1971 selon deux modes d'analyse :

- l'analyse dans un **contexte de territorialité** (figure V.1.a)
- l'analyse dans un **contexte de continuité** (figure V.1.b)

L'indice 100 qui sert de référence, correspond au revenu moyen des citoyens de l'Italie.

Représentation cartographique du P.N.B. par habitant en Italie en 1971

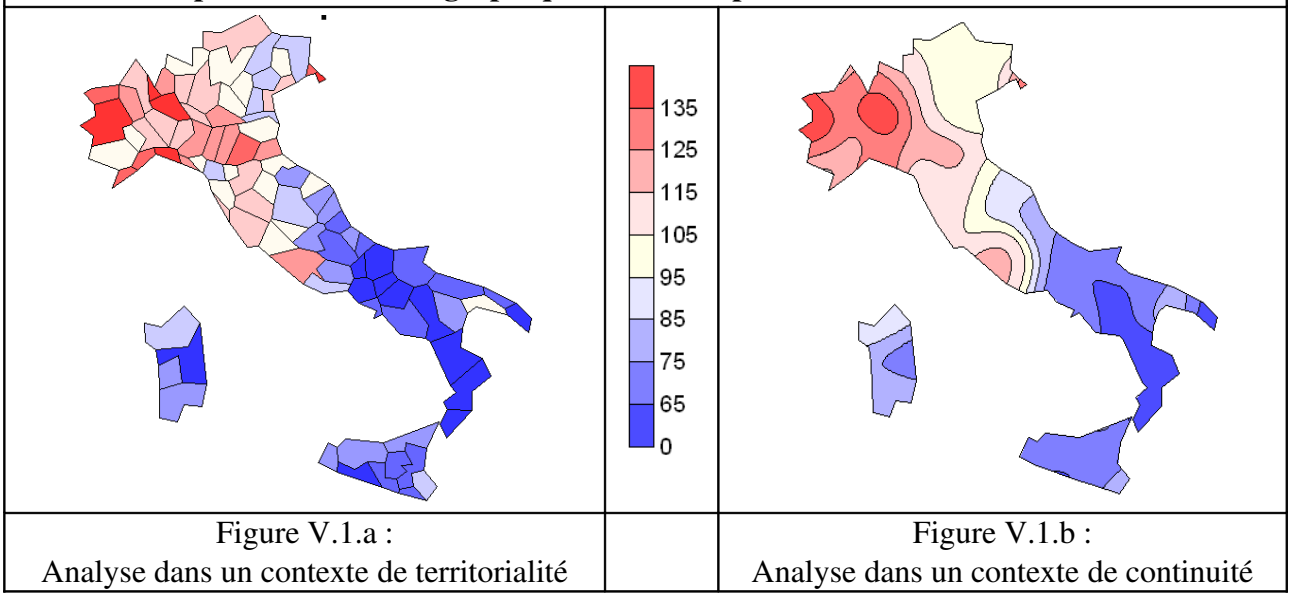


Figure V.1.a :
Analyse dans un contexte de territorialité

Figure V.1.b :
Analyse dans un contexte de continuité

Figure V.1 : Représentation cartographique suivant 2 modes d'analyse [GRASL99].

Remarque : seules les représentations cartographiques nous importe dans cet exemple. On ne cherche pas à connaître comment elles sont réalisées.

V.2 Spécifications cartographiques

Les spécifications cartographiques ont été proposées par l'équipe du laboratoire Géographie-cités. Elles prennent en compte les interactions possibles avec l'utilisateur. Les spécifications sont synthétisées et proposées par famille de cartes.

- **Carte représentant l'espace et les maillages** (carte des espaces)

Cette famille de carte permet de représenter l'espace de travail et les maillages que l'on souhaite utiliser pour notre analyse.
Le choix de l'espace d'étude devra faire apparaître en grisé la couverture des entités qui n'appartiennent pas à cet espace. Tandis que les entités contenues dans cet espace d'étude seront représentées avec les maillages sélectionnés.

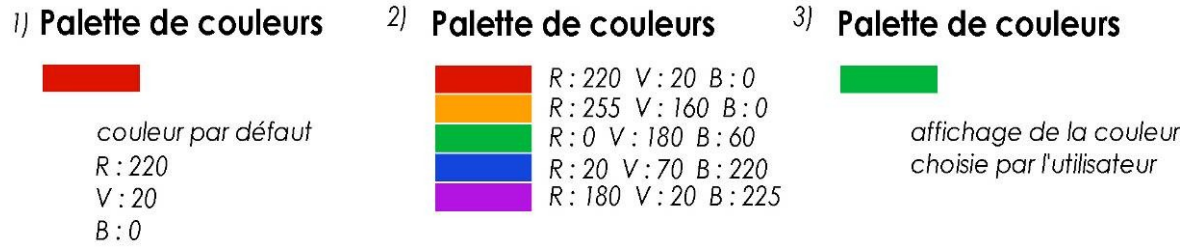
- **Carte représentant les variables** (carte des stocks)

La représentation de la carte des variables se fera à l'aide de cercles proportionnels par leur surface, à la valeur de l'attribut à représenter.

Il existe autant de cartes que de variables sélectionnées. Pour notre prototype les variables PIB et population seront sélectionnées grâce à deux listes déroulantes pour représenter deux cartes indépendantes.

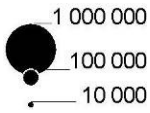
Les cercles qui se superposent (dans la légende et sur la carte) doivent être détourés. On affichera d'abord les plus gros cercles, puis les plus petits par dessus. Les cercles seront pleins et peuvent être de n'importe quelle couleur à partir d'une gamme proposée selon le stock à représenter.

Exemple *(figure ci-dessous) : il faut proposer à l'utilisateur d'autres couleurs de cercles que celle par défaut (ici le rouge) grâce à un bouton « palette de couleurs » (1). L'utilisateur visualise les couleurs à sa disposition et peut en choisir une en cliquant dessus (2). Celle-ci apparaît alors sous/à côté du bouton « palette de couleurs » (3).*



Principales contraintes sur la taille des cercles :
La taille des cercles maximal, minimal et « médian » de la légende doit être visible et de diamètre constant pour le stock maximum (2 valeurs de stock maximum pour 2 cartes distinctes sont représentées par un cercle de même dimension).

Exemple *de légende possible :*

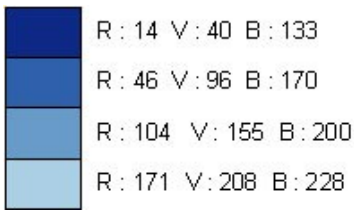


Lors d'un zoom, les cercles de la légende et de l'image ainsi que l'échelle graphique seront mis à l'échelle automatiquement. Autant que possible, la surface totale des cercles qui occupent la carte ne doit pas dépasser 10% de la surface totale de la surface de l'espace d'étude.

- **Carte représentant le ratio des deux variables** (carte de ratio)

Le ratio des deux variables sera représenté par une carte, suivant des calculs de quantiles[51], à l'aide d'aplats de couleurs dans une gamme de couleurs « simple[52] ».

Exemple d'une gamme « simple » bleue de 4 classes :



Edition de la légende par l'utilisateur :
L'utilisateur devra pouvoir choisir le nombre de classes (entre 2 et 10 classes et par défaut 4). Il aura le choix du type de quantile (quantile d'unités comme choix par défaut, quantile calculé sur le stock1 et quantile calculé sur le stock2).

[51] Le quantile est défini comme des classes de même effectif.
[52] Une gamme de couleur simple représente différentes couleurs données avec une dominante (le bleu par exemple) par opposition avec une gamme double qui comportera 2 gamme de couleurs simple (rouge et bleu par exemple).

Par exemple, pour la carte du PIB/hab., si l'utilisateur choisit une discrétisation en quatre classes :

- si le quantile d'unité est sélectionné, le calcul sera effectué sur le ratio PIB/habitant et chaque classe devra rassembler un quart de l'effectif total.
- si le quantile est calculé à partir du premier stock (PIB), chaque classe devra rassembler un quart de la richesse totale
- si le quantile est calculé à partir du second stock, chaque classe devra rassembler un quart de la population totale.

Remarque : ce sera toujours le ratio PIB/habitant qui sera représenté sur la carte.

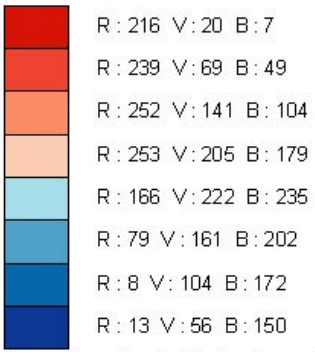
Une gamme de couleurs « simple » sera proposée à l'utilisateur. Il pourra éventuellement inverser la couleur de la gamme, la plus claire devenant la plus foncée et inversement.

Propriété du panneau légende :
Le titre de la légende doit reprendre le rapport de stock (e.g. PIB/habitant) et sa signification (e.g. exprime la richesse par habitant).
Une échelle graphique sera représentée.
La légende sera réalisée à partir de caissons jointifs suivant la gamme de couleurs sélectionnée. Elle portera l'indication des valeurs au niveau des bornes des caissons.

• Cartes représentant les déviations hiérarchiques

On représentera les deux cartes de déviations hiérarchiques à partir des indices correspondant au ratio des variables (avec indice 100 = moyenne des unités de l'espace concerné). La carte sera affichée à l'aide d'**aplats dans une gamme de couleurs « double »[53].**

Exemple d'une gamme Rouge Bleue « double » avec 8 classes :



Propriété de l'éditeur de légende :
L'utilisateur pourra choisir le nombre de classes à représenter (il est compris entre 2 et 10 et par défaut, 8 classes). Il choisira le type de progression (arithmétique ou géométrique, par défaut géométrique). Il aura le choix d'une gamme de couleurs « double » à sa disposition pour représenter les entités sur la carte. Il pourra aussi inverser cette gamme.

Propriété du panneau légende :
Le titre de la légende devra reprendre le rapport de stock (PIB/habitant) et la signification de l'indice 100 (indice 100= PIB/habitant européen, indice 100= moyenne des nuts2).
La légende est réalisée à l'aide d'aplats de couleurs avec caissons jointifs où l'on doit retrouver l'indication des valeurs au niveau des bornes des caissons. Une échelle graphique doit être intégrée.

[53] Ce choix de palette de couleurs n'apparaît pas dans la maquette de l'interface présentée.

Si l'indice 100 n'est pas borne de classe, il faut lui faire correspondre une classe de couleur jaune neutre.

La symétrie des couleurs dans les doubles gammes doit toujours être respectée (symétrie entre les rouges et les bleus). Par exemple, si on doit représenter 5 classes dans le positif et 4 classes dans le négatif, il faut se référer à la gamme de couleurs en 10 classes (5 positives et 5 négatives) et supprimer la dernière classe négative. Ne pas utiliser la gamme positive en 5 classes et la gamme négative en 4 classes (dans ce cas, la double gamme ne serait pas symétrique).

- **Carte représentant la déviation locale** (carte de voisinage)

La carte de déviation locale est représentée suivant ses indices (avec l'indice 100 correspondant à la moyenne des unités du maillage de référence contiguës à l'unité traitée), à l'aide d'aplats dans une gamme de couleurs « double ».

- **Carte représentant la synthèse des déviations** (carte de synthèse)

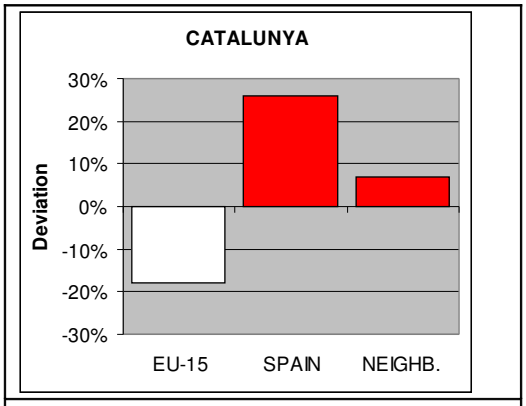
La carte représentant la synthèse des déviations établie une typologie combinant les déviations aux deux échelons supérieurs (échelon de l'espace d'étude et échelon du territoire de référence) et la déviation au voisinage. Elle sera réalisée à partir d'une gamme de couleurs double. La légende devra expliciter sans ambiguïtés les différentes déviations retenues.

V.3 Sortie graphique locale

Pour toutes les cartes, lors d'un « clic » de souris sur une unité territoriale (sa représentation géographique à l'écran), un histogramme apparaîtra dans une nouvelle fenêtre autonome contenant :

- Les informations propres à l'unité (les variables sélectionnées et le ratio correspondant)
- Les différentes déviations dans l'espace considéré (Europe des 15, pays et voisinage)

Exemple d'unité sélectionnée, en l'occurrence la Catalogne comparée à son voisinage (NEIGH), son Pays (SPAIN) et l'Europe des 15 (EU-15) :



Population	: 6 091 000 inh.
G.D.P.	: 91 110 000 000 euros
GDP/inh.	: 14 958 Euros/inh.

Sortie graphique locale (unité sélectionnée)

VI BIBLIOGRAPHIE

[ABPSU] D. Adair, J. Ball, M. Pawlan - tutorial Java2D - *http://java.sun.com/docs/books/tutorial/2d/*

[ALBRE95] J. Albrecht, Universelle GIS Operationen. Dissertation. Hochschule Vechta, Univ. de Vechta (1995)

[APACH] Projet Apache : *http://www.inria.fr/recherche/equipes/apache.fr.html*

[ATM03] C. Grasland, L. Lizzi, Multiscalar Territorial Analysis – 3rd version, European Spatial Planning Observatory Network, Annex of the second Interim Report of the TPG ESPON 3.1, 5 p. http://www.ums-riate.com/mta.php

[BARBI02] P. Barbier – Documentation texhnique Map Info V5.5 - livret 2- ENSG/ Cersig - IGN- ENSG-2002 - CERSIG - (2002)

[BEGUI94] M. Beguin M., D. Pumain, La représentation des données géographique Armand Colin, « Cursus3, Paris, 31 p. 1994

[BORLA] JBuilder, http://www.borland.fr/jbuilder/index.html

[CONAL00] Jim Conallen, Concevoir des applications Web avec UML. ED. Eyrolles. (2000)

[DAO03] H. Dao, Cours de Système d'Information Géographique - Université de Genève - Département de géographie - 46 p. (2003) http://www.unige.ch/ses/geo/cours/sig/welcome.html

[DENEG96] J. Denègre et f. Salgé, Les systèmes d'information géographique, Que sais-je ? – (1996)

[DEVEL] *http://developpeur.journaldunet.com/news/011110_sondage.shtml*

[DOREI99] M. Doreille M. Athapascan-1 : Vers un modèle de Programmation Parallèle adapté au calcul Scientifique. PhD thesis, Institut National Polytechnique de Grenoble, (1999)

[ESPON] European Spatial Planning Observation Network - *http://www.espon.lu*

[FLSVG] G. Held, T. Ullrich, A. Neumann, A. M. Winter – Comparing .SWF (ShockWave Flash) and .SVG (Scalable Vector Graphics) file format specifications - revised and corrected version, (2004) *http://www.carto.net/papers/svg/comparison_flash_svg/*

[FORMA] Formats, *http://fr.selfhtml.org/graphisme/formats.htm*

[GAMMA95] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns : Element of Reusable Object-Oriented Software", ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES (1995)

[GEORE] Portail géomatique - le carrefour francophone des disciplines composant la géomatique - *http://georezo.net/*

[GLEYZ01] A. M. Gleyze , Mise en place d'un système d'information sur les risques naturels. Etudes méthodologiques préalables- Mémoire de DEA Réseau « Structures et Dynamiques Spatiales » SEIGAD-IGA, Grenoble, Cemagref, (2001)

[GMMPV04] S. Gonzales, B. Marechal, L. Moragues, L. Puente, A. Vessot - La Cartographie sur Internet : Etat de l'art et aide au choix d'une solution - Micro-projet Mastère SILAT – Institut des sciences de l'homme – (2004)

[GOLCA01] F. Golay, R. Caloz – Traitements de l'information spatiale – Ecole Polytechnique Fédérale de Lausane – Laboratoire de système d'information géographique – 35 p. (2001) *http://lasig.epfl.ch/enseignement/sirs1.shtml*

[GRASL00] C. Grasland – Intérêt d'une approche multiscalaire – Cours de licence de géographie, Université Paris VII – (1999-2000)
http://www.grasland.cicrp.jussieu.fr/grasland/GO312/cours_d/cours_d1/cours_d1.html

[GRASL99] C. Grasland & al. - SPATIAL HOMOGENEITY AND TERRITORIAL DISCONTINUITIES - Application to the distribution of GDP/inh in Italy (1951-1991) - SPEPS, Working group 1.4, Working Paper n°2. (1999).
http://www.parisgeo.cnrs.fr/cg/spesp/wp2/index.htm

[HALL01] M. Hall, L. Brown - Core Web Programming (2nd Edition) - The Sun Microsystems Press, Java Series. (2001).

[HORST99] Cay S. Horstmann, G. Cornell, « Au cœur de Java 2 » volume1, Campus Press, (1999)

[HTML] HTML, *http://www.w3.org/MarkUp/*

[HYPERC] C. Grasland Historique détaillé du projet Hypercarte avec des exemples de réalisations. *http://www.parisgeo.cnrs.fr/cg/hyperc/index.htm*

[IDC04] Part de marché : *http://www.idc.fr* et *http://www.daratech.com/press/2003/030807/*

[IGNEN] Le serveur éducatif de l'IGN et de l'éducation nationale sur l'information géographique *http://pse.ensg.ign.fr/*

[MATPI02] H. Mathian, M. Piron, Echelles géographiques et méthodes statistiques multidimensionnelles (2003).

[MOTET97] S. Motet - Algorithmes et traitements Géométriques au DEA Sciences de l'information géographique. Mise à disposition sur le site de l'ENSG (*http://www.ensg.ign.fr*) rubrique "Développement programmation". 22 p. 45p. (1997)

[MULLE97] P.A. Muller, Modélisation objet avec UML, Eyrolles, 1997.

[ONGE95] B. Onge - Manuel hypertexte sur les structures de données appliquées aux Systèmes d'Information Géographique - Cours universitaire complet offert par l'Université du Québec à Montréal - Département de géographie – 1995 - *http://www.unites.uqam.ca/dgeo/geo7530/index1.htm*

[ROSE] Rational Rose, *http://www.rational.com/products/rose/*

[SCHOL02] M. Scholl - Modèles de représentation et d'interrogation de l'informatique géographique -*http://cedric.cnam.fr/vertigo/Cours/Valeur-C/spatial.pdf* (2002).

[SOUSS01] G. Soussain - Publier des cartes sur Internet : synthèse des différentes méthodes – Ministère de l'équipement, des transports et du logement – (2001)

[SPHER99] « DESCRIPTION OF WORK" Systematic, Palaeoflood and Historical data for the improvEment of flood Risk Estimation. PROPOSAL Nº: EVG1-1999-00039

[SVG] SVG, *http://www.w3c.org/Graphics/SVG/Overview.htm8*

[UML] UML, *http://www.omg.org*

[UNIGE] UNIGE, *http://www.unige.ch/ses/geo/cours/sig/welcome.html*

[W3C] World Wide Web Consortium, *http://www.w3.org*

[XML] XML, *http://www.w3.org/XML/*

MEMOIRE D'INGENIEUR C.N.A.M. en INFORMATIQUE

Interface cartographique pour l'analyse territoriale multiscalaire de phénomènes sociaux

Philippe Martin **Grenoble, le 14 décembre 2004**

Résumé :

Fournir sur le Web à des utilisateurs distants, des outils efficaces d'aide à la décision est une des opportunités de l'Internet. Le module d'Analyse Territoriale Multiscalaire (ATM), développé dans le cadre du projet Hypercarte, répond à cette attente dans le domaine de la cartographie. Il vise à établir des cartes observant la distribution spatiale de phénomènes sociaux selon un ou plusieurs maillages territoriaux.

Le module intègre une approche de type multiscalaire. En effet, les analyses hors contexte, souvent présentées dans les documents européens, mènent à une vision territoriale partielle de l'information. Ainsi, l'étude des différenciations sociales dans le cadre d'un maillage territorial permettra de représenter un même phénomène (la densité de population, la richesse par habitant, etc.) de façon très différente en fonction des hypothèses sur l'articulation des territoires administratifs (écarts d'une région par rapport au niveau européen, au niveau national, au niveau des régions voisines, etc.).

Le module est doté d'une interface conviviale permettant des requêtes utilisateurs visuelles pour obtenir de nombreuses familles de cartes. Son environnement cartographique interactif lui permet de procéder à des traitements classiques divers, comme le zoom, l'affichage du nom de l'entité sélectionnée, mais aussi plus évolués comme, par exemple, l'affichage contextuel d'histogrammes représentant trois déviations. De plus, il offre de nombreuses possibilités de personnalisation pour l'utilisateur comme le choix du nombre de classes à considérer pour les discrétisations, le type de progression (arithmétique ou géométrique), les couleurs à prendre en compte, etc.

La proposition formulée, apportée ici comme compréhension de phénomènes discrets via la production d'indicateurs couramment utilisés et reconnus, au sein d'une interface conviviale et interactive accessible à travers le Web, est une première étape pour l'analyse de phénomènes dans un monde en perpétuel changement.

Mots-clés :
Applet, Système d'information géographique, Aménagement du territoire, Interface graphique, Java, Modélisation UML, Web, Multimedia.

Keywords :
Applet, Geographical Information systems, Territorial information systems, Graphical interface, java, UML Modelling, Web, Multimedia.