CP 2I
DO
M éditerranée

Composants DESCARTES
Version 3.0 – Phase 1

Dossier de spécification V-3

Avril 2010

Ressources, territoires et habitats
Ressources, territoires, habitats et logement
Énergie et climat
Développement durable
Prévention des risques
Infrastructures, transports et mer

Présent pour l'avenir

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

Ministère de l'Écologie, de l'Énergie, du Développement durable et de la Mer

Centre de Prestations et d'Ingénierie Informatiques

Historique des versions du document

Version	Date	Auteur	Commentaires
1	15/03/2010	Denis CHABRIER (MEEDDM – CP2I – DO Méditerranée)	Première version basée sur : • les composants V2.0 • leur utilisation dans CARTELIE • le modèle commun de cartes
2	18/03/2010	Denis CHABRIER (MEEDDM – CP2I – DO Méditerranée)	Modifications après relecture interne au DO Méditerranée.
3	09/04/2010	Denis CHABRIER (MEEDDM – CP2I – DO Méditerranée)	Modifications après première relecture par le groupe « Modules » de la CCIG Mise à jour du planning

Documents de référence

Document	Version	Date	Rédacteur
Composants Descartes V2.0 - Manuel d'utilisation	2.1	15/02/10	D.CHABRIER
Modèle commun de cartes sur le Web	4	08/02/10	P.LAGARDE

Liste des destinataires

Nom	Organisme	Pour...
Clément JAQUEMET	MEEDDM / CGDD / DRI / MIG	Validation
Grégory MOREAU	MEEDDM / SPSSI / PSI 1	Validation
Christophe ROUSSON	MAAP / SDSI / BMSQ / DIG	Validation
Pierre LAGARDE	BRGM / STI/DIR	Validation
Christophe BOCQUET	MEEDDM / CP2I / DO Ouest	Validation
Olivier PELOUX	BULL Méditerranée	Mise en œuvre

Table des matières

1 Introduction

1.1 Présentation de la version en « production »

1.1.1 Objectifs et contenu

L'infrastructure du ministère pour répondre au besoin de « Cartographie sur Internet » repose principalement sur l'utilisation des normes publiées par l'Open Geospatial Consortium (OGC).

Afin de faciliter la mise en œuvre de ces normes dans les applications du ministère, DESCARTES propose aux équipes de maîtrise d'œuvre technique des composants logiciels directement utilisables.

Ceux-ci sont de deux types :

- Une bibliothèque JavaScript pour embarquer un « visualiseur OGC » dans une page Web
- Des services centralisés destinés à être inclus dans une application ACAI

Si le visualiseur JavaScript et les services côté serveur sont destinés à être utilisés conjointement, il est envisageable de les mettre en œuvre individuellement :

- utiliser le visualiseur sans les services côté serveur
- utiliser le visualiseur avec d'autres services côté serveur
- utiliser un autre visualiseur avec les services côté serveur

Dans les deux derniers cas, il convient toutefois de respecter le protocole de communication défini par DESCARTES.

Le visualiseur OGC

Il permet d'offrir à l'utilisateur l'essentiel des fonctionnalités habituellement souhaitées lors d'une consultation de cartes, tout en gérant l'ensemble des communications OGC requises.

Il s'agit de plus d'un client OGC autorisant l'accès simultané à plusieurs serveurs OGC. Une carte peut en ce sens être composée de couches issues de sources multiples.

La version actuelle (version 2) du visualiseur est basée sur des bibliothèques JavaScript reconnues dans la communauté du libre :

- prototype : extension du langage JavaScript et gestionnaire de requêtes AJAX
- scriptaculous : effets DHTML (seul le slider est utilisé)
- openLayers (version 2.7) : gestionnaire de couches géographiques

La bibliothèque « descartes.js » est chargée de piloter ces différents composants. Elle se compose d'un ensemble de classes permettant d'implémenter les spécificités fonctionnelles requises dans une application.

Les appels à toutes ces bibliothèques doivent être confiés à un code JavaScript « client » spécifique à chaque application.

Ce code « client » peut bien sûr utiliser d'autres bibliothèques. Dans le cas de Cartelie par

exemple, l'accordéon de la bibliothèque « rico » permet de gérer l'affichage alternatif de la plupart des gestionnaires (« contrôle des couches », « légende », « localisation rapide », etc.).

Les services centralisés

Certaines opérations proposées par les normes OGC ne peuvent, pour des raisons techniques, être mises en œuvre directement dans un navigateur Web par le langage JavaScript.

Cinq services centralisés sont proposés pour y pallier dans la version 2 :

- Gestion de requêtes WMS/GetFeatureInfo multiples avec réponses HTML
- Gestion de requêtes WFS/GetFeature multiples avec réponses HTML
- Gestion de requêtes WMS/GetFeatureInfo unitaires avec réponses XML
- Génération d'un fichier PNG à partir de requêtes WMS/GetMap
- Génération d'un fichier PDF à partir de requêtes WMS/GetMap

Ils sont disponibles pour des serveurs d'applications JAVA et PHP et donc aisément utilisables dans les applications ACAI.

1.1.2 Diffusion et accompagnement

La version actuelle des composants est mise à disposition sur le portail du développement du MEEDDM (Intranet).

Elle l'est de plus sur l'Adullact (forge des collectivités territoriales) depuis janvier 2010, avec une licence libre (CeCILL-C 2.0) : https://adullact.net/projects/descartes/

Outre les sources des composants, sont mises à disposition :

- une documentation comportant à la fois un volet « Guide de Référence » et un volet « Manuel de l'utilisateur »
- une application « exemple », illustrant la majeure partie des fonctionnalités offertes et disponible en version Java et Php.

Ces deux éléments d'accompagnement sont considérés comme majeurs pour obtenir l'adhésion aux composants par les maîtres d'œuvre.

1.2 Plan d'évolution envisagé

1.2.1 Éléments de contexte

Mutualisation interministérielle

Historiquement, DESCARTES provient de l'ancien ministère de l'équipement.

Les bouleversements des toutes dernières années ont été nombreux :

- nouveau périmètre ministériel (ex équipement, ex écologie, une partie de l'industrie)
- fusion en département des DDE (Ministère de l'équipement) et DDAF (Ministère de l'agriculture) pour donner lieu aux DDEA, qui seront elles-mêmes remplacées en 2010 par les DDT
- fusion en région des DRE (Ministère de l'équipement), des DIREN (Ministère de l'écologie) et d'un partie des DRIRE (Ministère de l'industrie)

Ces restructurations ont, pour le domaine de la cartographie sur le Web, conduit à mettre en concurrence plusieurs produits :

- CARTELIE (basé sur DESCARTES pour la consultation) de l'ex équipement
- CARMEN de l'ex écologie
- GEOWEB de l'agriculture
- PRODIGE, plateforme régionale

Dans un souci de limiter les effets pervers de cette concurrence et de mutualiser les efforts à fournir à l'avenir, une Commission de Coordination de l'Information Géographique (CCIG) a été créée entre le Ministère de l'écologie, de l'énergie, du développement durable et de la mer (MEEDDM) et le Ministère de l'alimentation, de l'agriculture et de la pêche (MAAP).

Dans le champ d'action de DESCARTES, les décisions d'ores-et-déjà prises par la CCIG sont les suivantes :

- les fonctionnalités de visualisation / consultation des différents produits doivent être prioritairement traitées par les mutualisations futures.
- le futur composant de visualisation / consultation mutualisé devra, dans la mesure du possible, offrir les fonctionnalités des produits actuels.
- la version actuelle des composants DESCARTES est la souche de base du futur composant
- la maîtrise d'ouvrage du projet est assurée par la sous-direction PSI du MEEDDM. La maîtrise d'œuvre est confiée au CETE Méditerranée.

Extension du périmètre fonctionnel

Actuellement, l'objectif des composants est ciblé sur le « grand public » et les utilisateurs internes non spécialistes de l'information géographique.

Les fonctionnalités proposées couvrent ainsi les actions de visualisation, consultation et interrogation.

Deux champs nouveaux sont désormais jugés prioritaires, chacun d'entre eux ayant une cible particulière :

- pour les utilisateurs internes spécialistes de l'information géographique (chargés d'études notamment), il est nécessaire de pouvoir modifier en ligne le style des objets d'une couche dans une carte, et plus particulièrement d'effectuer ces modifications selon des analyses statistiques paramétrables
- pour les applications « métier », il est indispensable de pouvoir gérer, côté client, la géométrie des objets géographiques manipulés

1.2.2 Typologie des évolutions

Intégration de bibliothèques JavaScript

L'intégralité de la version 1 des composants a été réalisée en interne par le CETE Méditerranée.

Dans le souci de faciliter la maintenance et l'évolution, la version 2 a été basée sur la bibliothèque OpenLayers pour la gestion cartographique pure (appels des couches, interactions de navigation).

Les widgets associés à une carte (gestionnaires des couches, des légendes, des recherches, etc.) ont toutefois continué à être réalisés en interne.

Pour la future version, ces widgets devront autant que possible reposer sur des bibliothèques JavaScript reconnues dans la communauté du libre. Sous réserves de tests d'aptitude à mener, ceux-ci devraient être la bibliothèque jQuery et ses compléments (jQueryUI, Dynatree, etc.)

Adaptation aux besoins des applications existantes

Dans le cadre de la mutualisation interministérielle, la future version des composants devra évidemment inclure les fonctionnalités présentes dans les applications existantes mais actuellement absentes de DESCARTES.

Il n'est toutefois pas envisagé d'être exhaustif : le groupe technique « Modules » de la CCIG a en charge d'identifier les fonctionnalités génériques pertinentes.

Symbolisation en ligne

Cela consiste à proposer à l'utilisateur, de manière individuelle (donc côté client), de modifier le style graphique des objets affichés, couche par couche.

Les modifications d'un style doivent être possibles sur toutes les caractéristiques de celui-ci (taille, couleur, épaisseur, symbole, etc.)

Pour un couche donnée, les modifications peuvent être de plusieurs types :

- globales : tous les objets de la couche ont un nouveau style identique
- différenciées suite à analyse statistique (plages, valeurs, symboles proportionnels, etc.)

Gestion des objets géographiques

Grâce au navigateur Web, l'utilisateur doit pouvoir interagir sur la géométrie des objets :

- création d'un nouvel objet
- modification d'un objet existant
- suppression d'un objet existant

Les créations et modifications peuvent le cas échéant être réalisées par rapport à d'autres objets (accrochage, fusion, scission, etc.).

Cette évolution a d'ores-et-déjà fait l'objet par le CETE Méditerranée d'une définition d'architecture technique respectant le modèle en couches d'ACAI.

L'architecture envisagée, qui a donné lieu à un prototype, repose sur des bibliothèque Java reconnues : Struts2, Java Topology Suite (JTS) et Hibernate Spatial.

Contrairement aux autres types d'évolutions, les composants serveurs ne seront réalisés a priori qu'en version Java.

1.2.3 Phasage

Afin d'éviter un « effet tunnel », il est envisagé de scinder le projet d'évolution des composants DESCARTES en deux phases se chevauchant partiellement.

La première phase, dite « phase Composants convergents », comportera les types d'évolution suivants :

- Intégration de bibliothèques JavaScript
- Adaptation aux besoins des applications existantes

La seconde phase, dite « phase Composants évolutifs », comportera les types d'évolution suivants :

- Symbolisation en ligne
- Gestion des objets géographiques

Le présent document est relatif à la première phase.

2 Spécifications fonctionnelles

La dimension « Composants » de l'opération nécessite de décrire les spécifications fonctionnelles de Descartes selon deux axes complémentaires :

- le service final devant être rendu aux utilisateurs d'une application cartographique intégrant les composants
- la richesse des possibilités d'adaptation, de configuration et de personnalisation des composants offerte aux maîtres d'œuvre des dites applications

2.1 Fonctionnalités pour les utilisateurs finaux

Les fonctionnalités décrites ci-après sont celles disponibles pour les utilisateurs finaux d'une mise en œuvre particulière des composants.

Bien évidemment, toutes ne seront pas offertes par l'ensemble des mises en œuvre : il convient à chaque maître d'œuvre de ne mettre à disposition que celles correspondant au périmètre fonctionnel souhaité.

De la même manière, pour une même fonctionnalité, plusieurs mode d'accès peuvent être disponibles en terme d'IHM : le maître d'œuvre doit donc choisir celle qui est offerte aux utilisateurs finaux.

Les fonctionnalités sont réparties en six catégories correspondant aux macro cas d'utilisation suivants :

- naviguer dans la carte
- modifier le mode de construction de la carte
- obtenir des informations générales sur la carte
- interroger les données présentées par la carte
- conserver des instantanés de la carte
- personnaliser l'interface cartographique

2.1.1 Naviguer dans la carte

Cette catégorie regroupe toutes les fonctionnalités permettant de modifier l'emprise de visualisation de la carte.

Déplacement de la zone de visualisation

Il s'agit ici de modifier l'emprise de visualisation de la carte sans changer l'échelle courante.

Méthodes possibles		Éléments IHM d'accès possibles
F-N1	recentrage suite à un clic sur la carte	rien (cf. « OpenLayers »)
		bouton/pictogramme « croix »
F-N2	glisser/relacher sur la carte	rien (cf. « OpenLayers »)
		bouton/pictogramme « main »
F-N3	translation dans une direction	croix « OpenLayers » (4 directions)
		« rose des vents » en bord de carte (8 directions)

F-N4	translation, dans une mini-carte de localisation, de la représentation de l'emprise courante	mini-carte « OpenLayers »
F-N5	saisie d'un couple de coordonnées	double zone de saisie textuelle

Changement d'échelle de visualisation

Il s'agit ici de modifier l'emprise de visualisation de la carte en changeant l'échelle courante, ce changement pouvant selon les cas être combiner avec un déplacement (cas des zoom « fenêtre » notamment).

Méthodes possibles		Éléments IHM d'accès possibles
F-N6	choix dans une liste d'échelle	curseur « OpenLayers » (horizontal ou vertical)
		liste déroulante textuelle
F-N7	saisie de l'échelle souhaitée	zone de saisie textuelle
F-N8	zoom avant simple	curseur « OpenLayers » (horizontal ou vertical)
		bouton/pictogramme « loupe + »
F-N9	zoom avant par dessin d'un rectangle	touche « shift » (cf. « OpenLayers »)
		bouton/pictogramme « loupe + »
F-N10	zoom arrière simple	curseur « OpenLayers » (horizontal ou vertical)
		bouton/pictogramme « loupe - »

Localisation directe

Il s'agit ici de se localiser directement sur une emprise sans interagir avec un quelconque outil cartographique (zoom, panoramique, etc.).

Méthodes possibles		Éléments IHM d'accès possibles
F-N11	sur l'emprise maximale de la carte	bouton/pictogramme « Tout voir »
F-N12	sur l'emprise initiale de la carte	bouton/pictogramme « Vue initiale »
F-N13	sur l'emprise précédente ou suivante de l'historique de navigation	boutons/pictogrammes « Précédent / Suivant »
F-N14	sur un élément d'un ensemble d'objets de référence	listes déroulantes textuelles imbriquées (ex : région / départements / communes)
F-N15	sur un objet issu d'une interrogation des données de la carte (voir 2.1.4 : Interroger les données présentées par la carte)	bouton/pictogramme « localiser »
F-N16	sur une emprise mémorisée (voir 2.1.5 : Conserver des instantanés de la carte)	bouton/pictogramme « Recharger la vue »

2.1.2 Modifier le mode de construction de la carte

L'ensemble de ces fonctionnalités est lié à l'existence d'un « gestionnaire de contrôle des couches ».

Ce gestionnaire présente sous une forme arborescente les couches de la carte pouvant donner lieu aux modifications décrites ci-après. L'arborescence est bien sûr de type « système de

fichiers », ce qui permet de proposer les actions de « pliage/dépliage » usuelles et d'avoir une profondeur non limitée.

Deux comportements exclusifs sont possibles :

- l'arborescence reflète fidèlement l'ordre de superposition des couches
- l'arborescence est déconnectée de l'ordre de superposition des couches

Gestion des couches affichées dans la carte

Il s'agit ici de modifier les certains paramètres des couches de la carte.

Fonctions possibles		Éléments IHM d'accès possibles
F-M1	Afficher / masquer une couche de la carte	bouton/pictogramme de type bascule à quatre états (oui / non / besoin zoom + / besoin zoom -)
F-M2	Régler l'opacité d'une couche de la carte	curseur de réglage
F-M3	Afficher / masquer un groupe de couches de la carte	bouton/pictogramme de type bascule à trois états (oui / non / mixte)
F-M4	Rendre une couche interrogeable ou non	bouton/pictogramme de type bascule à trois états (oui / non / inaccessible)

Organisation des couches dans une arborescence

Il s'agit ici de modifier l'organisation de l'arborescence des couches.

Ces fonctions ne sont possibles que si l'ordre de superposition des couches est cohérent avec l'arborescence.

Fonctions possibles		Éléments IHM d'accès possibles
F-M5	Ajouter / retirer une couche de la carte	À l'initiative de chaque application (par exemple par interfaçage avec un client CSW)
F-M6	Réordonner les couches au sens d'un groupe	???
F-M7	Déplacer les couches d'un groupe à un autre	???
F-M8	Réordonner les groupes de couches entre eux	???
F-M9	Ajouter / supprimer un groupe de couches	???

2.1.3 Obtenir des informations de la carte

Cette catégorie regroupe à la fois les fonctionnalités permettant d'offrir des éléments informatifs sur le contenu de la carte, autres que ceux correspondant aux objets des couches.

Informations propres aux couches

Fonctions possibles		Éléments IHM d'accès possibles
F-O1	visualiser les légendes des couches	automatique selon couches visibles et mode d'accès à l'initiative de chaque mise en œuvre
F-O2	accéder à des informations complémentaires (fiche de métadonnées, document associé...)	bouton/pictogramme d'accès à une URL

Informations cartographiques

Fonctions possibles		Éléments IHM d'accès possibles
F-O3	visualiser les coordonnées du curseur et le nom de la projection associée	zone de texte automatique
F-O4	visualiser l'échelle courante	échelle graphique automatique
		échelle métrique automatique
F-O5	visualiser les dimensions « terrain » de l'emprise courante	zone de texte automatique

Mesures sur la carte

Fonctions possibles		Éléments IHM d'accès possibles
F-O6	mesurer des distances par dessin de polylignes	bouton/pictogramme « Distances »
F-O7	mesurer des surfaces (et éventuellement des périmètres) par dessin de polygones	bouton/pictogramme « Surfaces »

2.1.4 Interroger les données présentées par la carte

Cette catégorie regroupe toutes les fonctionnalités permettant d'accéder aux données attributaires des couches constituant la carte.

Interrogations par l'intermédiaire de la carte

Il s'agit ici de rechercher les objets correspondant à une zone particulière de la carte.

Méthodes possibles		Éléments IHM d'accès possibles
F-I1	clic simple sur la carte	bouton/pictogramme « Sélection par simple clic »
F-I2	dessin d'un rectangle de sélection sur la carte	bouton/pictogramme « Sélection par rectangle »
F-I3	dessin d'un cercle de sélection sur la carte	bouton/pictogramme « Sélection par cercle »
F-I4	dessin d'un polygone de sélection sur la carte	bouton/pictogramme « Sélection libre »
F-I5	survol de la carte (info-bulle)	Aucun
F-I6	« hotlink » (liens sur zones sensitives)	Aucun

Le résultat des recherches de type F-I1 à F-I4 peut être affiché dans une fenêtre « pop-up » ou dans une zone de la page courante. Celle-ci peut comporter un bouton/pictogramme pour exporter le résultat au format CSV.

Pour les méthodes F-I2 à F-I4, ce résultat peut de plus comporter pour chaque objet trouvé un bouton/pictogramme « Localiser »

Autres interrogations

Il s'agit ici de rechercher des objets, sans interaction avec la carte.

Méthodes possibles		Éléments IHM d'accès possibles
F-I7	par saisie de critères de sélection pour une couche (requête attributaires)	zones de saisie en nombre variable, de type saisie libre ou liste de valeurs
F-I8	par consultation directe de l'ensemble des données d'une couche	bouton/pictogramme « Voir les données »

Le résultat des recherches peut être affiché dans une fenêtre « pop-up » ou dans une zone de la page courante. Celle-ci peut comporter un bouton/pictogramme pour exporter le résultat au format CSV. Ce résultat peut de plus comporter pour chaque objet trouvé un bouton/pictogramme « Localiser »

Pour la méthode F-I7, une couche de sélection est ajoutée à la carte pour repérer les objets satisfaisant les critères de recherche. Cette couche de sélection peut être supprimée de la carte grâce à un bouton/pictogramme « Effacer la sélection ».

2.1.5 Conserver des instantanés de la carte

Cette catégorie regroupe toutes les fonctionnalités permettant de sauvegarder, au sens large du terme, un état particulier de la carte au cours de la session de visualisation.

Création de documents à partir de la carte

Il s'agit ici d'exporter la carte sous forme de fichier externe.

Fonctions possibles		Éléments IHM d'accès possibles
F-C1	enregistrer la carte sous forme d'image	bouton/pictogramme « Enregistrer la carte »
F-C2	composer un document PDF à partir de la carte, des légendes, ...	bouton/pictogramme « Mise en page PDF » + assistant de mise en page

Gestion de contextes de visualisation

Il s'agit ici de proposer une gestion de vues personnalisées pour revenir à loisir sur des emprises / compositions jugées pertinentes par l'utilisateur.

Fonctions possibles		Éléments IHM d'accès possibles
F-C3	Mémoriser un contexte (emprise courante, taille de la carte, couches affichées avec leur opacité)	bouton/pictogramme « Enregistrer la vue »
F-C4	Supprimer un contexte mémorisé	bouton/pictogramme « Supprimer la vue »

2.1.6 Personnaliser l'interface cartographique

Cette catégorie regroupe toutes les fonctionnalités permettant de personnaliser l'interface générale de visualisation de la carte.

Fonctions possibles		Éléments IHM d'accès possibles
F-P1	Choisir la taille de la carte	liste déroulante textuelle
F-P2	Afficher / masquer la mini-carte de navigation	bouton/pictogramme « OpenLayers »

2.2 Fonctionnalités pour les maîtres d'œuvre

Les fonctionnalités décrites ci-après sont celles disponibles pour les utilisateurs des composants, c'est-à-dire les maîtres d'œuvre intégrant les composants dans une application.

Elles ont pour objectifs de configurer l'interface cartographique, au sens large, offerte aux utilisateurs finaux de la-dite application, selon le périmètre fonctionnel souhaité pour celle-ci.

A ce titre, les informations techniques nécessaires au fonctionnement des composants (URLs des serveurs WMS par exemple) ne sont pas décrites ici.

Cette configuration peut être répartie en quatre catégories :

- configurer les propriétés générales de la carte
- configurer le contenu des éléments de l'interface
- configurer les éléments IHM d'accès aux fonctions
- configurer l'ergonomie graphique

2.2.1 Configurer les propriétés générales de l'interface

Cette catégorie regroupe les possibilités de choix des fonctionnalités offertes par les composants en fonction du périmètre fonctionnel souhaité.

Modalités de navigation cartographique

Il s'agit de permettre au maître d'œuvre de choisir les fonctionnalités de navigation offertes à l'utilisateur final parmi celles disponibles dans le composant.

Choix de configuration		Possibilités de configuration
C-G1	Méthodes offertes	Sélection des méthodes nécessaires • recentrage suite à un clic sur la carte • glisser/relacher sur la carte • translation dans une direction • translation via une mini-carte de navigation • saisie d'un couple de coordonnées • choix dans une liste d'échelles • saisie de l'échelle souhaitée • zoom avant simple • zoom avant par dessin de rectangle • zoom sur l'emprise maximale de la carte • zoom sur l'emprise initiale de la carte • zoom sur l'emprise précédente ou suivante
C-G2	Mode de progression	Choix exclusif entre : • progression uniquement sur une liste d'échelles discrètes • progression libre, éventuellement bornée par une échelle maximale et/ou une échelle minimale

Pour C-G2, si la progression est basée sur une liste d'échelle discrètes, il n'est pas possible d'offrir la méthode de saisie d'un couple de coordonnées pour C-G1.

Informations d'aide à la lecture de la géographie

Il s'agit de permettre au maître d'œuvre de choisir les fonctionnalités d'information

cartographique offertes à l'utilisateur final parmi celles disponibles dans le composant.

Choix de configuration		Possibilités de configuration
C-G3	« Écouteurs » géographiques	Activation des « écouteurs » nécessaires parmi : • afficheur des coordonnées courantes • afficheur de la projection de la carte • afficheur de l'échelle graphique • afficheur de l'échelle métrique • afficheur des dimensions « terrain »
C-G4	« Mesureurs » géographiques	Activation des « mesureurs » nécessaires parmi : • distances par dessin de polylignes • surfaces par dessin de polygones
C-G5	« Mesureur » de surfaces	Activation ou non de l'affichage complémentaire pour le périmètre

Interactions non cartographiques

Il s'agit de permettre au maître d'œuvre de choisir les fonctionnalités autres que cartographiques offertes à l'utilisateur final parmi celles disponibles dans le composant.

Choix de configuration		Possibilités de configuration
C-G6	Modes d'interrogations des données	Sélection des modes nécessaires parmi : • interrogation ponctuelle • interrogation par rectangle • interrogation par cercle • interrogation par polygone • survol de la carte • consultation globale et directe des couches
C-G7	Modes de localisation indirecte	Sélection des modes nécessaires parmi : • suite à un choix dans un ensemble d'objets de référence • suite à un choix d'objet issu d'une interrogation des données
C-G8	Fonctions de paramétrage des couches	Sélection des fonctions nécessaires parmi : • affichage / masquage des couches et des groupes • réglage de l'opacité des couches • activer / désactiver une couche pour les interrogations • modification de l'ordre de superposition des couches • modification de l'ordonnancement de l'arborescence des couches (y compris groupes) • ajout / suppression de couches et de groupes
C-G9	Modes de sauvegarde de la carte	Sélection des modes nécessaires parmi : • enregistrement sous forme d'image • composition de document PDF • gestionnaire de vues personnalisées
C-G10	Accès à la modification de la taille de la carte	Activation du sélecteur de tailles

2.2.2 Configurer le contenu des éléments de l'interface

Cette catégorie regroupe les configurations nécessaires à l'alimentation des composants en contenu sémantique.

Élément de l'interface		Configuration nécessaire
C-C1	Constitution de la carte	Liste des couches de la carte avec leurs propriétés respectives.
C-C2	Gestionnaire de contrôle des couches	Liste des groupes de l'arborescence avec leurs propriétés respectives.
		Liste des couches d'un groupe avec leurs propriétés respectives.
C-C3	Sélecteur d'échelle	Liste des échelles de visualisation proposées
C-C4	Gestionnaire de requêtes attributaires	Liste des requêtes disponibles
		Liste des critères d'une requête
C-C5	Infos-bulles	Liste des couches concernées
		Liste des attributs concernés pour une couche
C-C6	Objets de référence	Niveaux d'imbrication avec leurs propriétés respectives
		Paramètres de l'éventuel service de fourniture des objets

2.2.3 Configurer les éléments IHM d'accès aux fonctions

Cette catégorie regroupe les possibilités de choix IHM pour accéder aux fonctionnalités permises, dans le cas où plusieurs options sont disponibles pour le maître d'œuvre.

Choix de configuration		Possibilités de configuration
C-A1	Accès au recentrage suite à un clic	Choix exclusif entre : • comportement par défaut d'OpenLayers • bouton / pictogramme « croix »
C-A2	Accès au glisser / relacher sur la carte	Choix exclusif entre : • comportement par défaut d'OpenLayers • bouton / pictogramme « main »
C-A3	Accès à la translation dans une direction	Choix exclusif entre : • « croix » d'OpenLayers • zones sensibles en bord de carte
C-A4	Accès au choix de l'échelle dans une liste	Choix exclusif entre : • « curseur » d'OpenLayers • liste déroulante textuelle
C-A5	Accès au zoom avant simple	Choix exclusif entre : • « curseur » d'OpenLayers • bouton / pictogramme « loupe + » et simple clic
C-A6	Accès au zoom avant par rectangle	Choix exclusif entre : • comportement par défaut d'OpenLayers (touche « shift ») • bouton / pictogramme « loupe + » et dessin de rectangle
C-A7	Accès au zoom arrière simple	Choix exclusif entre : • « curseur » d'OpenLayers • bouton / pictogramme « loupe - »

2.2.4 Configurer l'ergonomie graphique

Tous les éléments de l'IHM doivent pouvoir être aisément configurés au sens « rendu visuel » :

- boutons/pictogrammes
- textes informatifs
- zones de saisie libre
- listes déroulantes
- tableaux
- etc.

3 Spécifications techniques

3.1 Socle normatif, conceptuel et technique

3.1.1 Prépondérance des normes OGC

La mise en œuvre des normes OGC en matière de Web-mapping dans les composants doit être recherchée en priorité, et cela en prenant en compte les différentes versions de ces normes.

En cas d'impossibilité ou de difficulté technique avérée, les solutions de « contournement » proposées par le prestataire doivent être motivées.

Correspondances fonctionnalités / normes OGC

Le tableau suivant détaille les correspondances mises en œuvre dans la version 2.0 des composants :

Fonctionnalité		Norme / Opération
Afficher une couche		WMS / GetMap
Interrogation par clic		WMS / GetFeatureInfo
Interrogation par rectangle		WFS / GetFeature + FE
Interrogation par cercle		WFS / GetFeature + FE
Interrogation par polygone		WFS / GetFeature + FE
Info-bulle au survol		WMS / GetFeatureInfo
Affichage des données		WFS / GetFeature
Requête attributaire	Tableau des données	WFS / GetFeature + FE
	Sélection sur la carte	WMS / GetMap + SLD + FE

Ces correspondances doivent être respectées dans la version 3.0.

Incompatibilités fonctionnalités / normes OGC

L'impossibilité de mettre en œuvre grâce aux normes OGC certaines fonctionnalités sont d'ores-et-déjà identifiées. Il s'agit notamment de l'enregistrement sous forme d'image des cartes et de la composition d'un document PDF élaboré.

3.1.2 Adaptation du modèle commun de cartes

Un des objectifs du « modèle commun de cartes sur le web » est " *la définition d'un modèle permettant l'affichage d'une carte complexe sur Internet au travers d'un outil de cartographie dynamique* ".

La modélisation proposée comporte des informations dont le périmètre est plus large que celui couvert par les composants Descartes.

Par exemple, les métadonnées d'une carte (description, références temporelles, etc.), si elles ont éventuellement vocation à être indiquées dans l'interface générale d'une application, ne seront pas mises en œuvre directement par les composants mais par la page web (statique ou dynamique) embarquant les composants.

A contrario, la modélisation proposée ne fournit pas toutes les informations nécessaires à l'utilisation et à la configuration des composants.

Par exemple, les fonctionnalités décrites par la modélisation se limitent aux objectifs des interactions à proposer mais ne précisent pas les éléments d'IHM pour l'accès à celles-ci.

Le modèle de classes à élaborer lors de la conception technique, s'il doit être basé sur le « modèle commun de cartes sur le web », ne doit donc pas être un strict reflet de ce dernier.

3.1.3 Analyse de produits similaires

La version 2 de DESCARTES est bien évidemment la base de la version 3.

Néanmoins, tant pour l'implémentation des nouvelles fonctionnalités que pour les probables améliorations de l'existant, il est judicieux d'analyser certains produits similaires afin d'identifier dans ceux-ci les éléments qui pourraient éviter de redévelopper tout ou partie de ceux-ci. Il s'agit des produits suivants :

- l'API Géoportail
- le code Javascript de visualisation cartographique de CARMEN

Au sujet de l'API Géoportail, il est de plus indispensable d'assurer la compatibilité de DESCARTES avec la version minimale de celle-ci. L'accès aux services de données fournis par l'IGN doit en effet être efficient.

3.2 Architecture technique

L'architecture technique générale des composants doit reposer sur le principe du « client web riche », aussi connu sous le nom de « Rich Internet Application (RIA) ».

Ce principe n'est toutefois pas la panacée, la nécessité de traitements côté serveur étant indispensable dans plusieurs cas :

- impossibilité de réaliser les traitements souhaités pour des raisons d'incapacité du navigateur Web (génération de PDF par exemple)
- nécessité d'utiliser un serveur « proxy » pour pallier aux restrictions de sécurité du navigateur (« cross-domain » AJAX interdit)

Les composants doivent donc comporter :

- une bibliothèque JavaScript proposant le maximum des fonctionnalités requises
- des « services » centralisés proposant les autres fonctionnalités, pour des applications tant JAVA que PHP

3.2.1 Communication entre client / services

Les communications entre le client JavaScript et les services centralisés côté serveur doivent être mises en œuvre selon le respect des exigences suivantes :

- l'appel à un service doit être effectué selon une méthode HTTP de type POST, plus à même de transmettre des paramètres « hiérarchisés » et nombreux
- les paramètres d'appels à un service doivent être « publiés » et identiques quelque soit l'implémentation du service ; cela garantit l'éventuelle adaptation du client à des implémentations dans d'autres langages
- la réponse à l'appel d'un service doit pouvoir être fournie selon plusieurs formats selon le contexte IHM de récupération (par exemple : XML/GML brut, GeoJSON et HTML formaté)

3.2.2 Utilisation de frameworks

Bien évidemment, il ne s'agit pas de construire les composants ex-nihilo.

Des frameworks ou bibliothèques reconnus dans le monde du libre peuvent largement et efficacement contribuer à la mise en œuvre des spécifications fonctionnelles de Descartes.

Néanmoins, la mise à contribution de frameworks ou bibliothèques reconnus doit satisfaire les exigences suivantes :

- apporter une réelle plus-value : l'effet « sapin de Noël » doit en particulier être évité pour les IHM proposées par DESCARTES
- couvrir un champ fonctionnel restreint et précis : la situation où DESCARTES n'utiliserait qu'une proportion légère voir infime du périmètre fonctionnel offert par un framework ou un composant doit être évitée
- ne pas être directif vis-à-vis des applications : l'intégration de frameworks ou de composants ne doit pas remettre en cause l'architecture technique générale d'une application

De plus, les frameworks ou bibliothèques utilisées doivent impérativement être sous une licence compatible avec la mise à disposition de DESCARTES sous licence CeCILL-C 2.0.

Frameworks et bibliothèques JavaScript

La bibliothèque OpenLayers doit bien évidemment être la base de DESCARTES pour l'affichage d'une carte et les interactions de base liées à cette carte. C'est une librairie éprouvée et riche tant en couverture fonctionnelle qu'en capacité de personnalisation.

L'intégration d'autres bibliothèques ou frameworks est envisageable, et même souhaitable, dans les cas suivants :

- bibliothèques techniques telles que prototype.js ou jQuery facilitant le codage (pour les appels AJAX par exemple)
- bibliothèques « graphiques » telles que jQueryUI facilitant la création de widgets IHM

Frameworks et bibliothèques Java et PHP

L'intégration de frameworks ou de composants dans les services centralisés doit être motivée par l'impossibilité de rendre le service requis avec le langage de base utilisé.

A titre d'exemple, les composants côté serveur utilisés par la version 2.0 sont :

- commons-httpclient (et ses dépendances) pour les appels HTTP en Java
- iText pour la génération de PDF en Java
- fPdf pour la génération de PDF en Php

3.2.3 Configuration technique des composants

Dans l'objectif de faciliter l'intégration des composants dans une application, la configuration technique de ceux-ci doit à la fois consister en un nombre limité de paramètres et permettre leur adaptation au contexte IHM de l'application.

Configuration du client JavaScript

L'association du client à une IHM propre à l'application doit être réalisée uniquement par le biais d'un paramètre (ou d'une propriété de la méta-classe « DESCARTES ») indiquant le fichier CSS choisi.

Le code de programmation ne doit en aucune manière préciser autre chose que des références à des styles CSS, par l'intermédiaire des attributs HTML « id » ou « class ».

Bien évidemment, il est du ressort de chaque maître d'œuvre de constituer ce fichier CSS, ainsi que les pictogrammes associés à celui-ci.

Une IHM par défaut sera proposée dans le composant.

Par ailleurs, le composant devra offrir des méthodes pour paramétrer les URL d'accès aux services centralisés utilisés par le client JavaScript.

Configuration des services centralisés

Un nombre minimal de fichiers de configuration doit être recherché pour paramétrer les services centralisés.

Par exemple, pour la version JAVA, les paramètres seront concentrés dans « web.xml ».

3.3 Règles d'implémentation

Si toute application informatique doit satisfaire des exigences aidant à rendre son code lisible, maintenable et évolutif, ceci est renforcé dans le cas DESCARTES du fait de sa diffusion sur l'Adullact.

Les règles décrites dans ce chapitre doivent en conséquence être respectées pour concourir à cet objectif.

3.3.1 Codage JavaScript

Même si JavaScript n'est pas à proprement parler un langage de programmation orienté objet, ses capacités offrent la possibilité de créer par prototypage des pseudo-classes, et donc des objets instances de celles-ci.

Tout le code JavaScript doit être structuré de cette manière.

Organisation générale

Les sources doivent être organisées selon les règles suivantes :

- chaque classe est définie dans un fichier propre portant le nom de la classe
- un fichier ne définit qu'une seule classe
- une hiérarchie de classes est définie par un pseudo système de paquetages
- l'arborescence des fichiers sources reproduit la hiérarchie de classe
- la classe de base de l'arborescence est le méta-objet « Descartes »

Ces règles sont illustrées par les exemples suivants :

```
# Fichier « Racine » : Descartes.js »
var Descartes = Class.create()          // le méta-objet de base
Descartes.UnPremierPaquetage = Class.create();
Descartes.UnSecondPaquetage = Class.create();
...
# Fin du fichier

# Fichier « UnPremierPaquetage/UnePremiereClasse.js »
Descartes.UnPremierPaquetage. UnePremiereClasse = Class.create({
...
});
# Fin du fichier
```

```
# Fichier « UnPremierPaquetage/UneSecondeClasse.js »
Descartes. UnPremierPaquetage. UneSecondeClasse = Class.create({
...
});
# Fin du fichier

# Fichier « UnSecondPaquetage/UneDerniereClasse.js »
Descartes.UnSecondPaquetage. UneDerniereClasse = Class.create({
...
});
# Fin du fichier
```

Frameworks de définition des classes

Si le langage JavaScript se suffit à lui-même pour définir des classes, il est souvent préférable de se baser sur des frameworks facilitant le codage.

« prototype.js » est à ce titre performant et reconnu. Passer à une classe utilitaire un objet JSON stockant les propriétés et méthodes de la classe suffit à définir celle-ci :

Définition de classe avec prototype.js

```
UneClasse = Class.create({
	propriete1 : null,
	propriete2 : null,

	// le constructeur
	initialize : function(_prop1, _prop2) {
		this.propriete1 = _prop1;
		this.propriete2 = _prop2;
	},

	getPropriete1 : function() {
		return this.propriete1 ;
	}
});
```

OpenLayers propose lui-aussi une stratégie similaire à celle de prototype.js :

Définition de classe avec OpenLayers

```
UneClasse = OpenLayers.Class({
	propriete1 : null,
	propriete2 : null,

	// le constructeur
	initialize : function(_prop1, _prop2) {
		this.propriete1 = _prop1;
		this.propriete2 = _prop2;
	},

	getPropriete1 : function() {
		return this.propriete1 ;
	}
});
```

Une classe de DESCARTES peut donc être définie avec l'un ou l'autre des frameworks.

Toutefois, la définition avec prototype.js doit être préférée quand la classe n'hérite pas d'une classe OpenLayers. Cela permet de ne pas ajouter un couplage inutile entre DESCARTES et OpenLayers.

Par exemple, un gestionnaire de sélection dans un ensemble d'objets de références sera défini grâce à prototype.js car il peut éventuellement être utilisé dans un contexte non cartographique.

De la même manière, tous les appels XmlHttpRequest seront mis en œuvre via prototype.js.

Dans tous les cas, chaque classe doit posséder une propriété « CLASS_NAME » ayant pour valeur son nom complet dans le pseudo système de paquetage.

Par commodité, cette propriété sera toujours placée en fin de définition de la classe :

```
UnPaquetage.UnSousPaquetage.UneClasse = Class.create({
	propriete1 : null,
	propriete2 : null,

	initialize : function(_prop1, _prop2) {
		this.propriete1 = _prop1;
		this.propriete2 = _prop2;
	},

	getPropriete1 : function() {
		return this.propriete1 ;
	},
	CLASS_NAME: "UnPaquetage.UnSousPaquetage.UneClasse"
});
```

Cohérence avec les codes HTML et CSS

Certaines classes ont vocation à proposer à l'utilisateur final des éléments d'IHM.

Pour ce faire, la seule stratégie devant être implémentée consiste à lier une instance de classe à une DIV destinée à contenir les éléments générés, en passant au constructeur de la classe l'identifiant de la DIV.

L'exemple suivant illustre cette stratégie :

Définition de la classe (« UneClasse.js »)

```
UneClasse = Class.create({
	div : null,

	initialize : function(idDiv) {
		this.div = document.getElementById(idDiv);
	},

	draw : function() {
		this.div.innerHTML = "quelque chose";
	},
	CLASS_NAME : "UneClasse" ;
});
```

Utilisation de la classe dans une page HTML

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
	<head>
		<script src="prototype_1_6_0_1.js"></script>
```

```
            <script src="UneClasse.js"></script>
            <script>
                  function init() {
                        var unObjet= new UneClasse('info');
                        unObjet.draw();
                  }
            </script>
            <title>Exemple MVC</title>
      </head>
      <body onload="init()">
            <div id="info"></div>
      </body>
</html>
```

De plus, l'aspect graphique de chaque élément d'IHM généré est défini par des styles CSS personnalisables.

Pour associer un style personnalisé à un élément d'IHM généré par une classe, la seule stratégie devant être implémentée consiste à fournir au constructeur de la classe les noms des classes CSS à appliquer.

L'exemple suivant illustre cette stratégie :

```
UneClasse = Class.create({
      div : null,

      initialize : function(idDiv, displayClass) {
            this.div = document.getElementById(idDiv);
            this.div.className = displayClass;
      },

      draw : function() {
            this.div.innerHTML = "quelque chose";
      },
      CLASS_NAME : "UneClasse" ;
});
```

Constructeurs de classe avec objet JSON pour les options

La définition des signatures de méthodes dans le langage JavaScript ne suit pas véritablement les règles de l'art. Il est tout à fait possible d'exécuter une méthode sans passer l'intégralité des paramètres potentiellement acceptés.

Cela ne pose pas trop de problèmes quand le nombre de paramètres potentiels est faible. Par contre cela n'est pas adapté au cas de composants offrant de multiples options de configuration.

La signature d'un constructeur de classe ne doit donc comporter que les paramètres indispensables à une instanciation. Les paramètres optionnels doivent eux être passés en tant que propriétés d'un objet JSON structuré.

L'exemple suivant illustre cette stratégie :

```
UneClasse = Class.create({
      div : null,

      initialize : function(idDiv, options) {
```

```
            this.div = document.getElementById(idDiv);
            this.option1 = options.option1 || 0;
            this.option2 = options.option2 || "";
      },
      CLASS_NAME : "UneClasse" ;
});
```

Patterns de conception

Certains patterns de conception contribuent à la modularité, à la lisibilité et à l'extensibilité du code :

- la mise en place d'écouteurs (listeners).

  Une instance de classe s'abonne à un événement d'une instance d'une autre classe. Quand cet événement est déclenché par l'instance émettrice, l'instance de classe abonnée est notifiée et active ses traitements propres

- l'anonymat des fonctions de retour (callbacks).

  Une instance de classe ignore la ou les utilisations faites de ses propres traitements. Les fonctions responsables de ces utilisations sont passées en paramètres dans le constructeur de la classe, puis exécutées de manière anonyme.

- la modélisation MVC.

  Ce pattern, largement répandu, clarifie le code en « isolant » traitements, objets manipulés et représentation de ces objets dans l'IHM.

Ces patterns doivent, dans la mesure du possible, être respectés.

Des exemples de mise en œuvre de ceux-ci sont présentés en annexe.

Par ailleurs, l'ouvrage « Pro JavaScript Design Patterns » aux éditions Apress est une bonne référence pour structurer le code Javascript.

3.3.2 Codages Java et PHP

Les règles de l'art d'implémentation dans ces langages sont davantage nombreuses et connues que pour JavaScript. Elles ne font donc pas l'objet de spécifications particulières.

Toutefois, les implémentations des services centralisés doivent impérativement être compatibles avec les frameworks mis en œuvre dans les applications.

Il ne faut pas, par exemple, privilégier un quelconque framework MVC.

En conséquence, les implémentations doivent être le plus proches possibles du langage de base utilisé. Ainsi, pour la version Java, les services seront réalisés sous forme de servlets.

4 Livrables attendus

4.1 Conception technique

Le livrable correspondant est un Dossier de Conception Technique (DCT), éventuellement accompagnés d'un projet UML réalisé avec un logiciel de modélisation.

4.1.1 Conception générale

Le DCT contiendra obligatoirement les éléments suivants :

- la liste des frameworks et bibliothèques externes proposés (JavaScript, Java et Php), avec un argumentaire étayé pour chacun d'entre-eux

- la définition du protocole de communication entre le client JavaScript et les services centralisés. Pour chaque service centralisé, les éléments à définir sont les suivants :
  - liste des paramètres des appels HTTP
  - formats possibles des réponses aux appels
  - structurations des réponses pour chaque format

4.1.2 Conception des composants JavaScript

Le DCT contiendra obligatoirement les éléments suivants :

- l'organisation en pseudo système de paquetages de l'API JavaScript

- la définition de chaque classe de l'API JavaScript, suffisamment documentée pour en comprendre l'objectif et le fonctionnement. Les éléments à définir sont les suivants :
  - interface de la classe : nom, propriétés avec leur valeur par défaut, constructeur avec ses paramètres (y compris les paramètres optionnels), méthodes avec leurs paramètres et leur éventuel valeur de retour
  - information sur l'éventuel héritage : classe mère, méthodes surchargées

- la liste des interactions internes entre classes de l'API Javascript. Pour chaque classe, les éléments à définir sont les suivants :
  - déclenchements des méthodes des classes « propriétés »
  - abonnements à des événements d'autres classes
  - fonctions de retour fournies à d'autres classes

- la cinématique (diagrammes de séquence) de certaines actions déclenchées par l'utilisateur. Les actions concernées sont :
  - celles mettant en œuvre le pattern MVC dans le client JavaScript
  - celles mettant en œuvre à la fois le client JavaScript et les services centralisés

- la définition de la modélisation des éléments IHM (liste des DIV alimentées par les classes de l'API JavaScript, liste des classes CSS par défaut)

4.1.3 Conception des composants Java et Php

Le DCT contiendra obligatoirement les éléments suivants :

- la conception générale des services centralisés, pour les versions Java et Php. Les éléments à définir sont les suivants :
  - contraintes d'exploitation nécessaires vis-à-vis des éléments de plateforme pour les versions (JRE, Tomcat, Php...) et dépendances (modules de Php par exemple)
  - liste des paramètres de configuration et modalités de « stockage »
  - liste des threads nécessités par des temps de traitements incertains

- la définition de chaque classe des API Java et Php, suffisamment documentée pour en comprendre l'objectif et le fonctionnement. Les éléments à définir sont les suivants :
  - propriétés et méthodes publiques de la classe
  - information sur l'éventuel héritage de la classe : nom de la classe mère, méthodes surchargées

4.2 Composants logiciels

4.2.1 Sources

Les sources doivent être livrées selon les spécifications décrites précédemment (cf. 3.3.1 Codage JavaScript page 24) pour les composants JavaScript et selon les règles d'usages pour les composants Java et Php.

Les sources doivent être intégralement documentées de manière à pouvoir générer aisément la documentation des API. A ce titre, la complétude de la documentation doit aller jusqu'aux propriétés et méthodes privées.

Si pour les composants Java le formalisme retenu doit être naturellement JavaDoc, le prestataire proposera pour les composants JavaScript et Php des formalismes cohérents avec des générateurs de documentation reconnus et fiables. Par exemple :

- http://www.naturaldocs.org/ pour JavaScript et PHP
- http://code.google.com/p/jsdoc-toolkit/ pour JavaScript

Dans le cas des composants JavaScript, l'exigence de modularité des sources (un fichier par classe) rend de facto fastidieuse l'intégration de celles-ci dans une page HTML.

Afin de simplifier cette intégration, les sources des composants Javascript doivent être accompagnées d'un fichier contenant un script de chargement automatique, seul fichier à intégrer dans une page HTML.

Cette stratégie est d'ores-et-déjà mise en œuvre dans OpenLayers et dans la version 2.0 de DESCARTES.

4.2.2 Versions « compilées »

Si la diffusion finale des composants comportera bien évidemment les sources de ceux-ci, mise à disposition sur l'Adullact oblige, elle proposera aussi une version dite « compilée » pour les utilisateurs ne se préoccupant pas des vues internes.

Selon les langages, les possibilités de « compilation » sont les suivantes :

- Java : création d'une archive JAR
- JavaScript : fusion et compactage (grâce à l'utilitaire JSMIN) en un fichier unique

- Php : aucune possibilité

Par ailleurs, chaque fichier diffusé sur la forge de l'Adullact doit posséder les informations relatives à la licence d'utilisation propre à Descartes.

En conséquence, les sources doivent être accompagnées d'un script ANT pour générer les versions « compilées » et insérer le contenu d'un fichier décrivant la licence Descartes.

Ce script ANT inclura aussi la génération des documentations d'API.

4.3 Éléments d'accompagnement

4.3.1 Documentation

Le manuel d'utilisation des composants, destinés aux maîtres d'œuvre d'applications, doit comporter, a minima, les chapitres suivants :

- description exhaustive et sommairement illustrée de l'API JavaScript (y compris les éventuelles méthodes utilitaires du méta-objet Descartes)
- description exhaustive de l'IHM proposée par défaut (styles CSS et pictogrammes)
- modalité intégration des composants « client » dans une application (liens entre DIV, classes CSS et classes Descartes)
- description exhaustive du ou des protocoles de communication entre l'API JavaScript et les services centralisés
- installation et configuration des services centralisés
- présentation didactique d'une sélection des exemples d'utilisation

4.3.2 Exemples d'utilisation

Accompagnant le manuel d'utilisation, des exemples directement utilisables doivent présenter les différentes fonctionnalités offertes par les composants, tant pour les utilisateurs finaux que pour les maîtres d'œuvre.

Ces exemples doivent illustrer les capacités suivantes des composants :

- variété du périmètre fonctionnel offert à l'utilisateur final (5 exemples)
- variété des IHM offertes par simple instanciation des classes de l'API JavaScript (2 exemples)
- variété des IHM offertes par styles CSS et pictogrammes personnalisés (3 exemples)

Les exemples doivent être disponibles en deux versions :

- une application Java / Tomcat
- une application Php

5 Annexes

5.1 Organisation générale

5.1.1 Intervenants

Organisme ou instance	Rôle	Représentant
MEEDDM / SG / SPSSI / PSI 1	Maîtrise d'ouvrage	G. MOREAU
MEDDDM / CP2I / DO Méditerranée	Maîtrise d'œuvre générale	D. CHABRIER
CCIG	Validation des spécifications	Groupe « Modules »
MEDDDM / CP2I / DO Ouest	Maîtrise d'œuvre de recette	C. BOCQUET
BULL Méditerranée	Sous-traitant de conception et réalisation	O. PELOUX

5.1.2 Planning

Action	Responsable	Début	Fin	Charge
Prise en charge des composants	BULL Méditerranée	Mi mars	Fin mars	12 j.h
Rédaction des spécifications	CP2I / DO Méditerranée	Mi février	Mi mars	15 j.h
Validation des spécifications	CCIG - Groupe « Modules »	Mi mars	Mi avril	10 j.h
Réalisation de la conception technique	BULL Méditerranée	Mi avril	Mi mai	15 j.h
Validation de la conception technique	CP2I / DO Méditerranée	Mi mai	Mi mai	5 j.h
Réalisation des composants	BULL Méditerranée	Mi mai	Fin juin	40 j.h
Réalisation des éléments d'accompagnement	BULL Méditerranée	Fin mai	Fin juin	20 j.h
Recette des composants et des éléments d'accompagnement	CP2I / DO Méditerranée CP2I / DO Ouest	Début juillet	Fin juillet	25 j.h 25 j.h
Ajustements après les recettes	BULL Méditerranée	Début août	Fin août	10 j.h
Recette finale et diffusion	CP2I / DO Méditerranée	Fin août	Début septembre	3 j.h

La charge totale correspond à 180 J.H

5.2 Patterns de conception JavaScript : exemples

5.2.1 Écouteurs

Définition d'une classe abonnée à un événement

```
Control = new Object();
Control.GeoDims = Class.create({
      initialize: function(idDiv, map) {
            this.idDiv = idDiv;
```

```
            this.map = map;
            this.map.events.register( 'moveend', this, this.updateDims);
            this.updateDims();
      },
      updateDims: function() {
            $(this.idDiv).innerHTML = "Largeur : " + this.map.getExtent().getWidth()
                              + "<br>Hauteur : " + this.map.getExtent().getHeight();
      }
});
```

Utilisation de la classe dans une page HTML

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
      <head>
            <style>
                  .smallmap {width: 512px; height: 256px; border: 1px solid #ccc;}
            </style>
            <script src="prototype_1_6_0_1.js"></script>
            <script src="lib/OpenLayers.js"></script>
            <script src="ecouteur.js"></script>
            <script>
                  function init(){
                        var map = new OpenLayers.Map('map');
                        var layer = new OpenLayers.Layer.WMS( "OpenLayers WMS",
                                    "http://labs.metacarta.com/wms/vmap0",
                                    {layers: 'basic'}
                              );
                        map.addLayer(layer);
                        map.zoomToMaxExtent();

                        new Control.GeoDims('geodims', map);
                  }
            </script>
            <title>Exemple d'écouteur</title>
      </head>
      <body onload="init()">
            <div id="map" class="smallmap"></div>
            <div id="geodims"></div>
      </body>
</html>
```

5.2.2 Fonctions de retour

Définition d'une classe déclenchant des fonctions de retour

```
Calcul = Class.create({
      initialize: function(arg1, arg2, callbacks) {
            this.result = arg1 + arg2;
            this.callbacks = callbacks;
      },
      execute: function() {
            for (var i=0 ; i<this.callbacks.length ; i++) {
                  this.callbacks[i](this.result);
            }
      }
});
```

Utilisation de la classe dans une page HTML avec deux fonctions de retour

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
	<head>
		<script src="prototype_1_6_0_1.js"></script>
		<script src="retours.js"></script>
		<script>
			function init() {
				var monCalcul = new Calcul(1,1,[retour1, retour2]);
				monCalcul.execute();
			};
			var retour1 = function(texte) { alert(texte); };
			var retour2 = function(texte) { $('retour').innerHTML = texte; };
		</script>
		<title>Exemple de retours multiples</title>
	</head>
	<body onload="init()">
		<div id="retour"></div>
	</body>
</html>
```

5.2.3 Modélisation MVC

Définition d'un Contrôleur

```
Mvc.Controleur = Class.create({
	initialize: function (exempleModele, vueSaisie, vueInfo) {
		this.modele = exempleModele;
		var self= this;
		this.vueSaisie = new Mvc.VueSaisie(vueSaisie.idDiv, vueSaisie.templateHTML,
							this.modele, self);
		this.vueInfo = new Mvc.VueInfo(vueInfo.idDiv, vueInfo.templateHTML,
							this.modele);
	},
	activate : function () {
		this.vueSaisie.paint();
		this.vueInfo.paint();
	},
	actualiseModele : function () {
		// Recuperation des valeurs saisies
		var modeleSaisi = this.vueSaisie.getDatas();
		// Validation 'metier' des saisies
		if (modeleSaisi.hasValidDatas()) {
			// mise a jour du modele
			this.modele.setDatas(modeleSaisi);
			this.vueInfo.paint();
		} else {
			alert("Valeurs incorrectes");
		}
	}
});
```

Définition d'un Modèle

```
Mvc.Modele = Class.create({
	initialize: function (datas) {
		this.prop1 = datas.prop1 || 'défaut';
		this.prop2 = datas.prop2 || 'défaut';
```

```
            this.prop3 = datas.prop3 || 'fixe';
      },
      setDatas: function (modele) {
            this.prop1 = modele.prop1 || this.prop1;
            this.prop2 = modele.prop2 || this.prop2;
            this.prop3 = modele.prop3 || this.prop3;
      },
      hasValidDatas: function () {
            // Les regles de gestion...
            var isValid = (this.prop1 != null && this.prop2 != null)
                        && (this.prop1 != "" && this.prop2 != "");
            return isValid;
      }
});
```

Définition d'un Vue de saisie

```
Mvc.VueSaisie = Class.create({
      initialize: function (id, templateHTML, modele, controleur) {
            this.idDiv = id;
            this.modele = modele;
            this.controleur = controleur;
            this.template = new Mvc.TemplateEngine(templateHTML, id);
      },
      getDatas: function () {
            var modeleSaisi = new Mvc.Modele({});
            return this.template.sendDatas(modeleSaisi);
      },
      paint: function (contenu) {
            $(this.idDiv).innerHTML = this.template.render(this.modele);
            var self = this;
            $(this.idDiv + '_Actualiser').onclick =
                        function() {self.controleur.actualiseModele();};
      }
});
```

Définition d'une Vue d'affichage

```
Mvc.VueInfo = Class.create({
      initialize: function (id, templateHTML, modele) {
            this.idDiv = id;
            this.modele = modele;
            this.template = new Mvc.TemplateEngine(templateHTML, id);
      },
      paint: function (contenu) {
            $(this.idDiv).innerHTML = this.template.render(this.modele);
      }
});

// Le gestionnaire de Templates
Mvc.TemplateEngine = Class.create({...});
```

Utilisation des classes MVC dans une page HTML

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
      <head>
            <script src="prototype_1_6_0_1.js"></script>
            <script src="mvc.js"></script>
            <script>
```

```
function loadExample() {
      var leModele = new Mvc.Modele({prop1:'valeur 1', prop2:'valeur 2'});

      var templateSaisie = 'Propriété 1 : <input type="text" value="{prop1}"/>'
                           + '<br/>Propriété 2 : <input type="text" value="{prop2}"/>'
                           + '<br/><input type="button" value="#Actualiser#"/><br/>';
      var templateInfo = 'Propriété 1 : <span>{prop1}</span><br/>'
                           + 'Propriété 2 : <span>{prop2}</span><br/>'
                           + 'Propriété 3 : <span>{prop3}</span>';

      var configVueSaisie = {idDiv:'saisie',templateHTML: templateSaisie};
      var configVueInfo = {idDiv:'info',templateHTML: templateInfo};

      var leControleur = new Mvc.Controleur(leModele, configVueSaisie, configVueInfo);
      leControleur.activate();
};
            </script>
            <title>Exemple MVC</title>
      </head>
      <body onload="loadExample()">
            <div id="saisie"></div>
            <div id="info"></div>
      </body>
</html>
```

5.3 Glossaire

5.3.1 Organismes et instances

Sigle	Signification
CCIG	Commission de Coordination de l'Information Géographique
CETE	Centre d'Études Techniques de l'Équipement
CP2I	Centre de Prestations de d'Ingénierie Informatique
DDAF	Direction Départementale de l'Agriculture et des Forêts
DDE	Direction Départementale de l'Équipement
DDEA	Direction Départementale de l'Équipement et de l'Agriculture
DDT	Direction Départementale des Territoires
DIREN	Direction Régionale de l'ENvironnement
DO	Département Opérationnel
DRE	Direction Régionale de l'Équipement
DRIRE	Direction Régionale de l'Industrie de la Recherche et de l'Environnement
ED-GEO	Études et Développements en GEOmatique
MAAP	Ministère de l'Agriculture, de l'Alimentation et de la Pêche
MEEDDM	Ministère de l'Écologie, de l'Énergie, du Développement Durable et de la Mer
OGC	Open Geospatial Consortium

5.3.2 Technologies et acronymes

Sigle	Signification
ACAI	Architecture Commune des Applications Informatiques
AJAX	Asynchronous Javascript And Xml
CSS	Cascaded Style Sheets
CSV	Comma-Separated Values
CSW	Catalogue Service Web
DHTML	Dynamic HyperText Markup Language
FE	Filter Encoding
GeoJSON	Geographic JavaScript Object Notation
GML	Geographic Markup Language
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IHM	Interface Homme-Machine
JSON	JavaScript Object Notation
MVC	Model View Controlert
PDF	Portable Document Format
PNG	Portable Network Graphics
RIA	Rich Internet Application
SLD	Styled Layer Descriptor
WFS	Web Feature Service
WMS	Web Map Service
XML	eXtensible Markup Language