CONSERVATOIRE NATIONAL DES ARTS ET METIERS
CENTRE REGIONAL ASSOCIÉ DE TOULOUSE

MÉMOIRE

présenté en vue d'obtenir

le DIPLÔME d'INGÉNIEUR CNAM

SPÉCIALITÉ : INFORMATIQUE

OPTION : INFORMATIQUE, RÉSEAUX SYSTÈMES ET MULTIMÉDIA

par

GARDOLL Sébastien

Umodelis une plateforme de modélisation hydrologique

Soutenu le 02 juin 2010

JURY

PRESIDENT : POLLET Yann (PTC CNAM)

MEMBRES : BATATIA Hadj (MCF CNAM)
BONNET Marie-Paule (CR IRD)
COCHONNEAU Gérard (IR IRD)
PIOMBO Christophe (PrAG CNAM)

Remerciements

Les premières personnes que je tiens à remercier sont Marie-Paule Bonnet et Gérard Cochonneau, les instigateurs du projet Umodelis. Marie et Gérard, je vous remercie pour la confiance que vous m'avez témoignée pour la réalisation de ce projet, les compétences scientifiques et techniques que vous m'avez apportées. Votre aide a dépassé le cadre professionnel : je vous remercie sincèrement pour l'aide financière, celle relative à l'administration brésilienne, celle sur la difficile question du logement et du transport, j'en oublie bien entendu ...
Mes remerciements vont ensuite aux directeurs de l'IRD Brésil Pierre Sabaté et son successeur Jean-Loup Guyot qui m'ont accueilli dans leur délégation. Je remercie chaleureusement (abraço) Valdemar Santos Guimarães et Eurides de Oliveira qui m'ont intégré dans leur équipe : vos marques d'attentions et votre hospitalité m'ont touché. J'ai eu le plaisir de faire partie de la famille du personnel expatrié de l'IRD ainsi que celle de l'ANA qui m'a fait partager son art de vivre, confirmant ainsi la célèbre convivialité brésilienne. Je tiens à exprimer ma reconnaissance pour Geraldo Boaventura et Patrick Seyler ainsi que l'équipe administrative de la délégation IRD Brésil, Bertha Surlemont et Sacy Nadaradjane (et ses petits fours indiens), pour le montage du dossier du stage et autres tracas administratifs.
Ma gratitude va également à Jacqueline et Jacques Gardoll, Gérard, Yann Sivry, Sophie Soulans pour le temps consacré à débugger mon manuscrit.
Nadja, merci beaucoup pour ton hospitalité, ta gentillesse, tes conseils et ton aide déterminante pour la recherche de logement. Merci également à Laurent et Stéphanie pour leur amitié. A Laurent Durieux et Marie, je n'oublierai pas tous ces merveilleux souvenirs de colocation, nos soirées riches en discussions scientifiques et sociologiques sans oublier nos sorties bien animées ! J'éprouve, comme prévu, beaucoup de nostalgie de cette époque.
Ce travail étant le fruit d'un long parcours, j'aimerais remercier les personnes qui m'ont soutenu pendant toute l'époque des cours du soir à commencer par Sophie Soulans qui a su m'apporter un soutien indéfectible depuis le jour où j'ai poussé la porte du CNAM jusqu'à maintenant, tu as beaucoup compté dans cette reconversion. Yann Sivry et Laila Yahi, pour votre sincère amitié, votre soutien pendant la rédaction, merci mille fois.
Je remercie également Bernard Dupré qui a compris mon désir de reconversion et cru en mes capacités ainsi que mes anciens collègues du service salle blanche : Caco Boucayrand et Michel Valadon qui m'ont encouragé pendant toutes les années CNAM. J'ai également une pensée amicale pour mes autres anciens collègues Mireille Polvé, Bernard Rénier, Jean Claude Harrychourry, Pierre Brunet, Rémi Freydier, Fred Candaudap, Carole Causserand, Christelle Lagane, Manu Henry, Mickaël Autuori, pour tous ces moments passés en leur compagnie.
Enfin, j'aimerai exprimer ma profonde affection pour ma famille et le réconfort qu'elle m'a apporté pendant les périodes difficiles.

Toulouse, le 01 mars 2010

Abréviations

LISAH Laboratoire d'étude des Interactions Sol-Agrosystème–Hydrosytème

ADCP Acoustic Doppler Current Profiler

ANA Agência Nacional de Águas

API Application Programming Interface

ATHYS l'ATelier HYdrologique Spatialisé

BSD Berkeley Software Distribution

CASH Contribution de l'Altimétrie Spatiale à l'Hydrologie

CNAM Conservatoire National des Arts et Métiers

CNRS Centre National de la Recherche Scientifique

CORBA Common Object Request Broker Architecture

DIAS Dynamic Information Architecture System

DLL Dynamic Link Library

DoS Deny of Service

DSDM Dynamic Software Development Method

DIS Direction des Systèmes d'Information

EAR Enterprise ARchive

Etc. et cetera desunt

EIS Entreprise Information System

EJB Enterprise JavaBean

EPL Eclipse Public License

EPST Etablissement Public à caractère Scientifique et Technologique

ERD Entity-Relationship Diagram

ESRI Environmental Systems Research Institute

GIS Geographic Information System

GPL General Public License

GRASS Geographic Resources Analysis Support System

GUI Graphical User Interface

gvSIG generalidad valenciana SIG

HTTP HyperText Transfer Protocol

IHM Interface Homme Machine

IIOP Internet Inter-Orb Protocol

IRD Institut de Recherche pour le Développement

ISO International Organization for Standardization

J2EE Java 2 Enterprise Edition

JAMS Jena Adaptable Modelling System

JDBC Java Data Base Connectivity

JNDI Java Naming and Directory Interface

JNI Java Native Interface

JPEG Joint Photographic Experts Group

JSP Java Server Pages

JTS Java Topology Suite

JUMP Java Unified Mapping Platform

JVM Java Virtual Machine

LAN Local Area Network

LGPL Lesser General Public License

LMTG Laboratoire des Mécanismes et Transferts en Géologique

LTHE Laboratoire d'étude et des Transferts en Hydrologie et Environnement

MCD Modèle Conceptuel de Données

MDB Message-Driven Beans

MHYDAS Modélisation HYdrologique Distribuée des AgroSystèmes

MIMD Multi Instructions on Multiple Data

MLD Modèle Logique de Données

MNT Modèle Numérique de Terrain

MOM Message-Oriented Middleware

MPI Message Passing Interface

MVC Modèle Vue Contrôleur

OGC Open Geospatial Consortium

OMG Object Managment Group

OMP Observatoire Midi-Pyrénées

OMT Object Modeling Technique

OpenMI Open Modelling Interface

OpenMP Open Multi-Processing

ORB Object Request Broker

ORE-HYBAM L'Observatoire de Recherche en Environnement contrôles géodynamique, HYdrologique et biogéochimique de l'érosion/altération et des transferts de matières dans le bassin de l'AMazone

ORSTOM Office de Recherche Scientifique et Technique Outre-Mer

OSGeo Open Source Geospatial foundation

OSGi Open Services Gateway initiative

pH potentiel hydrogène
PK Primary Key
QGIS Quantum GIS
QoS Quality of Service
RAD Rapid Application Development
RCP Rich Client Platform (Eclipse RCP)
RMI Remote Method Invocation
RUP Rational United Process
SADT Structured Analysis and Design Technique
SDK Software Development Kit
SF-SQL Simple Feature – Structured Query Language
SGBDR Système de Gestion de Base de Données Relationnelle
SIG Système d'Information Géographique
SIMD Single Instructions on Multiple Data
SPMD Single Process on Multiple Data
SME Spatial Modelling Environment
SO Shared Object (Unix)
SQL Structured Query Language
STL Standard Template Library
Syn. Synonyme
SWT Standard Widget Toolkit
TIFF Tagged Image File Format
TIME The Invisible Modelling Environment
Trad. Traduction
UC Unité de Calcul
uDig user-friendly Desktop internet gis
UI User Interface
UML Unified Modeling Language
UNFAO United Nation Food & Agriculture Organization
UP Unified Process
UPS Université Paul Sabatier
URL Uniform Resource Locator
WAN Wide Area Network
WFS Web Feature Service
WMS Web Map Service
XML extensible Markup Language

Glossaire

ADCP (Syn. courantomètre)

Affluent (Trad. tributary, Syn. tributaire)
Voir tributaire.

Aire de drainage (Trad. drained area, Syn. surface drainée)
Surface « d'évacuation, par gravité ou par pompage, d'eaux superficielles ou souterraines dans une zone donnée. » (MUSY, 2005).

Analyse systémique
La compréhension d'un système complexe par son organisation et les interactions de ses composants.

Anthropie
Relatif à l'activité humaine.

Apache Tomcat
Containeur de servlet et de JSP maintenu par l'apache software foundation.

Applicatif (Trad. business logic, Syn. expertise métier et fonctionnalité)
Voir expertise métier.

Application d'entreprise (Trad. business service)

Application Web dynamique
Application accessible sur le Web, exécutée au travers d'un navigateur Web et dont le comportement évolue selon les interactions avec l'utilisateur.

Architecture distribuée (Syn. architecture n tiers et multi tiers)
Architecture où les ressources ne sont pas sur une même machine mais réparties sur plusieurs.

Architecture multi tiers (Syn. architecture distribuée et n tiers)
Voir architecture distribuée.

Architecture n tiers (Syn. architecture distribuée et multi tiers)
Voir architecture distribuée.

Balance de charge (Trad. load balancing)
Voir chapitre 11 page 83.

Bas niveau
Désigne un niveau d'abstraction de l'information proche du langage machine, en opposition à haut niveau.

Bascule d'urgence (Trad. failover)
Voir chapitre 11 page 83.

Bassin fluvial (Syn. bassin fluvial, Trad. watershed)
Voir bassin versant.

Bassin versant (Syn. bassin fluvial, Trad. watershed)
« Ensemble d'une région ayant un exutoire commun pour ses écoulements de surface.» (MUSY, 2005).

Bief (Trad. reach, Syn. tronçon)
Voir reach.

Boost
Bibliothèque d'utilitaires écris en C++ portable sur de multiples systèmes d'exploitation.

Bus de communication (Syn. ORB)
Lien entre les applications appartenant à des systèmes hétérogènes liés en réseau. Le bus de communication est au cœur de la communication d'un intergiciel en permettant l'invocation à distance de méthodes sur des objets distribués.

Business logic (Trad. expertise métier, applicatif et fonctionnalité)
Voir expertise métier.

Business service (Trad. application d'entreprise)
Expertises métier mises à disposition de tiers.

Calculs distribués (Syn. Calculs répartis)
Voir chapitre 11 page 83.

Calculs parallèles
Voir chapitre 11 page 83.

Calculs répartis (Syn. calculs distribués)
Voir chapitre 11 page 83.

Carte (Trad map)
Container de couches d'objets géographiques. Entité organisant l'information géographique.

Centroïde (d'une géométrie)
« Point fictif situé à l'intérieur d'un polygone » enveloppant la forme géométrique « et dont les coordonnées correspondent généralement au centre de ce polygone. » (terminologie informatique de l'office québécois de la langue française).

Cluster (Trad. Ferme de calcul)

Codage
Action d'exprimer l'information dans un dialecte donné.

Conductivité électrique de l'eau
Mesure la force ionique de l'eau, la concentration de toutes les espèces ioniques présentes.

Couche (d'objets géographiques) (Trad. layer)

Courantomètre (Syn. ADCP)
Appareil basé sur l'effet doppler servant à mesurer la vitesse d'un courant d'eau.

Déni de service (Trad. Deny of Service)

Deny of Service (Trad. déni de service)
« Impossibilité, pour des utilisateurs autorisés, d'accéder à des ressources ou introduction de retards dans l'exécution des opérations. » (terminologie informatique de l'office québécois de la langue française).

Distribué (Syn. Répartie)

Donnée cartographique
Terme générale désignant les cartes et les couches d'objets géographiques et de données spatialisées.

Donnée hydrologique
Dans Umodelis, désigne une donnée concernant l'hydrologie comme un débit d'eau ou une longueur de cours d'eau.

Donnée non spatialisée
Dans Umodelis, désigne une donnée qui n'est pas localisée par des coordonnées.

Donnée spatialisée
Donnée qui est localisable par ses coordonnées.

Drainage (bassin versant)
Ecoulement par gravité de l'eau au sein d'un bassin versant.

Eclipse RCP
Plateforme logicielle générique servant de base pour le développement d'applications riches graphiques.

Eléments en traces
Eléments sous forme ionique et de très faible concentration (10^{-9} g/L d'eau).

Eléments majeurs (Syn. Ions majeurs)

Enterprise ARchive
Format de fichier Java qui sert d'emballage à une application distribuée J2EE.

Entité de calcul
Terme générique référant aussi bien à une unité de calcul qu'à une ferme de calcul.

Expertise métier (Trad. business logic, Syn. applicatif et fonctionnalité)
Plus value d'un logiciel, ses fonctionnalités et services rendus.

Exutoire (d'un réseau hydrographique)
« Ouverture ou passage par lequel s'écoule le débit sortant d'un réservoir ou d'un cours d'eau. » ou « Point le plus bas d'un réseau hydrographique. » (MUSY, 2005).

Failover (Trad. balance d'urgence)

Ferme de calcul (Trad. cluster)
Ensemble d'unités de calcul interconnectées.

Fil d'exécution (Trad. thread, Syn. processus léger)
Voir thread.

Fonctionnalité (Trad. business logic, Syn. expertise métier et applicatif)
Voir expertise métier.

Gadgets graphiques (Trad. widgets)
Terme générique désignant les contrôles visuels (boutons, boîtes de dialogues, etc.) composant les IHM.

Géolocaliser (Syn. géoréférencer et spatialiser)
Voir géoréférencer.

Géomatique
« Discipline ayant pour objet la gestion des données géographiques et qui fait appel aux sciences et aux technologies reliées à leur acquisition, leur stockage, leur traitement et leur diffusion » (terminologie géomatique de l'office québécois de la langue française).

Géoréférencer (Syn. géolocaliser et spatialiser)
« Opération qui consiste à redresser la localisation relative des objets géographiques en les reportant dans un système de référence absolue. » (terminologie géomatique de l'office québécois de la langue française).

Grand compte (Trad. key account, Syn. compte clef)
Compte important sur le plan du chiffre d'affaire d'une entreprise.

Grid (Trad. raster)

GUI (Trad. IHM et interface graphique)
Voir IHM.

Haut niveau
Désigne un niveau d'abstraction de l'information proche de l'expression en langage naturel, en opposition à bas niveau.

Hydro-climatique
Relatif à l'hydrologie et au climat.

IHM (Trad. GUI, Syn. interface graphique)
Mécanismes graphiques de contrôles et d'actionneurs qui permettent l'interaction entre l'humain et la machine.

Image matricielle
Image formée d'une matrice de pixels. Si les pixels sont géréférencés, on parle alors de raster.

Image vectorielle
Image formée à partir de figures géométriques basiques comme le point, le segment, le polygone, etc. L'image vectorielle est orientée objet.

Implémentation
« Opération qui consiste à réaliser la phase finale d'élaboration d'un système, afin de le rendre fonctionnel. » (terminologie informatique de l'office québécois de la langue française). Soit un codage dans un langage informatique donné soit une réalisation d'une spécification, d'un modèle abstrait, etc.

Interface graphique (Trad. GUI, Syn. IHM)
Voir IHM.

Intergiciel (Trad. middleware, Syn. logiciel médiateur)
« Logiciel qui sert d'intermédiaire transparent entre des applications appartenant à des systèmes hétérogènes liés en réseau, lesquels entretiennent le plus souvent des relations basées sur le modèle client-serveur. » (terminologie informatique de l'office québécois de la langue française).

Intergiciel orienté messages (Trad. MOM)
Intergiciel basé sur l'échange de messages atomiques.

Interpolation de séries altimétriques
Calcul visant à compléter les données manquantes des séries de données sur l'altitude d'un objet géographique.

Interpolation spatiale
Estimation d'une information spatiale manquante à partir de l'information adjacente (site Web department

of geomatics, Melbourne university). Un exemple est la ligne par interpolation linéaire.

Ions majeurs
Les ions Cl^-, K^+, Mg^{2+}, SO_4^{2-}, F^-, Ca^{2+} et Na^+ en concentraion importante (g/L d'eau).

Itération
« Répétition d'une séquence d'instructions ou d'une partie de programme un nombre de fois fixé à l'avance ou tant qu'une condition définie n'est pas remplie, dans le but de reprendre un traitement sur des données différentes. » (terminologie informatique de l'office québécois de la langue française).

J2EE
Architecture n tiers développée par Sun Microsystème s'appuyant sur le protocole RMI-IIOP, voir chapitre 11 page 83.

JFace
Bibliothèque de contrôles graphiques avancés en Java basée sur SWT.

Layer (Trad. Couche (d'objets géographiques))
Container d'objets géographiques de même nature dans une logique de représentation graphique.

Leveling (Trad. nivellement)

Ligne par interpolation linéaire de points (Trad. line string)
Forme géométrique spatialisée modélisant une ligne et qui est composée de coordonnées. La représentation de la ligne entre les coordonnées est estimée par régression linéaire.

Limnigraphe
Appareil construit sur le principe de flotteur servant à mesurer la hauteur d'eau d'un cours d'eau en continu.

Line string
Trad. ligne par interpolation linéaire de points.

Linear ring (Trad. Anneau)

Load balancing (Trad. balance de charge)
Voir chapitre 11 page 83.

Macrocommande
« Séquence de commandes, de touches de fonction et d'instructions enregistrée sous un nom, qu'on peut rappeler et exécuter par une commande unique ou par le nom qui lui a été attribué. » (terminologie informatique de l'office québécois de la langue française).

Mandataire (Trad. proxy)
Entité recevant de la part d'un tiers le droit de le représenter. Dans un contexte distribué, un mandataire

représente un tiers à distance.

Map (Trad. carte)

Mapping (Trad. table de correspondance)
Table basée sur la correspondance entre une clef et une valeur associée.

Message atomique (Syn. opération atomique et cohérente)
Voir opération atomique.

Métadonnée (Description d'un type de données)

Middleware (Trad. intergiciel et logiciel médiateur)

MIMD
Machine à plusieurs unités de calcul travaillant indépendamment les uns des autres.

Modèle entité-relation
Modélisation décrivant un logiciel par les relations qu'entretiennent les entités du système. L'implémentation de ce genre de modèle la plus connue est MERISE.

Modèle hydrologique linéaire ou 1D
Modèle mathématique dont l'hypothèse est la linéarité de la réponse des entités hydrologiques modélisées.

Modèle mathématique
Une description mathématique approchée et paramétrée d'un phénomène réel.

Modèle Numérique de Terrain
Représentation numérique du relief d'une surface continentale.

Module (Trad. plugin)
Outil ou programme pouvant être ajouté à un outil ou à un programme principal par le biais d'un point d'extension.

MOM (Trad. intergiciel orienté messages)

Muskingum-Cunge
modèle mathématique de calculs de débits d'un cours d'eau par propagation (CUNGE, 1969).

Nivellement (Trad. leveling)
Différence d'altitude entre deux coordonnées.

Objet distribué
Objet (au sens OMT) dont les méthodes sont invocables à distance, à l'aide de mandataires, par les appli-

cations liées par un bus de communication.

Objet géographique
« Phénomène modélisé à des fins de représentation cartographique. » (terminologie géomatique de l'office québécois de la langue française). Terme générique représentant par exemple des stations hydrométriques, des biefs, des bassins versant, etc.

Objet hydrologique
Dans Umodelis, modélisation orientée objet d'une donnée hydrologique. Voir chapitre 8 page 47.

OGC
Consortium pour la coordination et la promotion des standards ouverts en géomatique.

Opération atomique (Syn. Message atomique et opération cohérente)
Opération qui ne peut être préemptée afin d'assurer la cohérence de son action sur le système.

Opération cohérente (Syn. Message atomique et opération atomique)
Voir opération atomique.

ORB (Syn. bus de communication)

OSGeo
Fondation pour la promotion des logiciels géomatiques libre de droit et open source.

Paralélisation
Voir chapitre 11 page 83.

Patron de conception (Trad. Design pattern)
Modèle de conception logicielle réutilisable répondant à un problème connu, popularisé par GAMMA *et al.* (1994).

Pédologie
Etude de l'organisation et la caractérisation des sols.

pH
Mesure de l'activité de l'hydrogène.

Plugin (Trad. module)

Plugin extention point
Trad. point d'extension (de module).

Pluie moyenne sur un bassin
Apport moyen d'eau sur un bassin versant provenant des précipitations.

Point d'extension (de module) (Trad. plugin extension point)
Entité spécifiant l'intégration de nouveaux modules.

Préempter
Sélection du ou des processus à exécuter ou dont l'exécution est à suspendre, au sein d'un système d'exploitation.

Processus
Tache exécutant une pile d'instructions machine et possédant son propre espace mémoire pour les données de travail.

Processus léger (Trad. Thread, Syn. fil d'exécution)
Voir thread.

Proxy (Trad. mandataire)

Rapport isotopique
Le quotient de la quantité d'un isotope d'un atome par rapport à celle d'un isotope différent du même atome.

Raster (Trad. grid)
Matrice de pixels juxtaposés pondérés d'une ou plusieurs valeurs numériques pouvant être représentées graphiquement (couleur, données hydrologiques avec une coordonnée, etc.).

Reach (Trad. tronçon et bief)
Un reach est une portion d'un cours d'eau dont les délimitations sont librement choisies.

Répartie (Syn. Distribué)

Réseau hydrographique (Trad. hydrographic network)
« Ensemble des rivières et autres cours d'eau permanents ou temporaires, ainsi que des lacs et des réservoirs, dans une région donnée. » (MUSY, 2005).

Ressource de calcul (Syn. unité de calcul)

Retour en arrière (Trad. rollback)

Rollback (Trad. retour en arrière)
Annule une transaction effectuée en remettant l'état du système dans son l'état avant l'exécution de la transaction.

Scheduler (Trad. ordonnanceur)
« Programme dont l'objet est d'allouer du temps de processeur aux tâches ou travaux prêts à être exécutés. » (terminologie informatique de l'office québécois de la langue française).

Serveur d'applications (Trad. application serveur)
« Serveur abritant les applications destinées à être utilisées dans un réseau distribué. » (terminologie informatique de l'office québécois de la langue française).

Servlet
Programme au sein de J2EE responsable du traitement, au niveau d'un serveur, d'une requête exprimée par un client Web.

SF-SQL
Langage de requêtes ensembliste basé sur le langage SQL et conçu pour de l'information spatialisée.

SIMD
Machine à plusieurs unités de calcul travaillant en parallèle.

Site hydrologique
Dans Umodelis, un site hydrologique représente la réunion d'objets géographiques et des données spatialisées ou non.

Spatialiser (Syn. géoréférencer et géolocaliser)
Voir géoréférencer.

Station hydrométrique
« Station où sont effectués des relevés sur un ou plusieurs des éléments suivants relatifs aux eaux des rivières, des lacs et des réservoirs : hauteur d'eau, débit, transport et dépôt de matériaux, température et autres propriétés physiques de l'eau, caractéristiques de la couverture de glace et propriétés chimiques de l'eau. » (MUSY, 2005).

Stream (Trad. Cours d'eau) **Surcouche d'emballage**
Couche qui sert à encapsuler une couche et à exprimer ses interfaces sous d'autres formes d'interfaces.

Surface drainée (Trad. drained area, Syn. aire de drainage)
Voir aire de drainage.

Système de projection (de coordonnées géographiques)
La Terre n'étant pas parfaitement sphérique, un système de projection est un modèle mathématique convertissant des coordonnées spatiales en coordonnées planaires afin de représenter des entités tri-dimensionnelles sur une carte (site Web department of geomatics, Melbourne university).

Télédétection spatiale
« C'est l'ensemble des connaissances et techniques utilisées pour déterminer des caractéristiques physiques et biologiques d'objets par des mesures effectuées à distance, sans aucun contact matériel avec ceux-ci » (journal officiel de la république française). « Le terme télédétection est couramment utilisé pour désigner la télédétection électromagnétique aérospatiale appliquée à l'étude de la surface et de l'atmosphère de la Terre

et, par extension, des planètes. » (terminologie géomatique de l'office québécois de la langue française).

Thread (Trad. fil d'exécution et processus léger)
Tache exécutant au sein d'un processus une pile d'instructions machine et qui utilise l'espace mémoire du processus pour ses données de travail. L'espace mémoire du processus est partagé entre tous les threads.

Thread périodique
Thread exécuté à chaque intervalle de temps (période) donné.

Thread SWT
Thread capturant les interactions entre l'utilisateur et l'interface graphique composée à l'aide de la bibliothèque SWT (à rapprocher avec le thread swing).

Trace satellitaire
Projection de la position d'un satellite sur la Terre à un instant donné.

Transaction
Une opération cohérente composée de plusieurs opérations unitaires dont l'intégrité de l'exécution est garantie et dans le cas contraire un retour en arrière (rollback) est possible.

Tributaire (Trad. tributary, Syn. affluent)
« Cours d'eau ou rivière qui se jette dans une rivière plus grande ou un lac. » (MUSY, 2005).

Tributary (Trad. tributaire et affluent)
Voir tributaire.

Tronçon (Trad. reach, Syn. bief)
Voir reach.

UML
Langage de modélisation logicielle proposant 13 diagrammes pour la description de la conception d'un logiciel orienté objet.

Unité de calcul (Syn. ressources de calcul)
Désigne un processeur capable d'effectuer un calcul.

Watershed (Trad. bassin versant)

Widgets (Trad. gadgets graphiques)

Wrapper (Trad. surcouche d'emballage)

Table des matières

Chapitre 1

Introduction

Ce mémoire s'inscrit dans l'étude hydrologique des bassins versants. Afin de comprendre son contexte scientifique, la notion de bassin versant sera abordée ainsi que l'intérêt de son étude avec comme exemple le bassin amazonien. Par la suite, les moyens d'investigation scientifiques seront présentés afin de comprendre la problématique posée par la communauté scientifique. Le chapitre se termine par une courte description des instituts qui ont permis ce travail et un guide de lecture du mémoire.

1.1 Études et observations des bassins versants

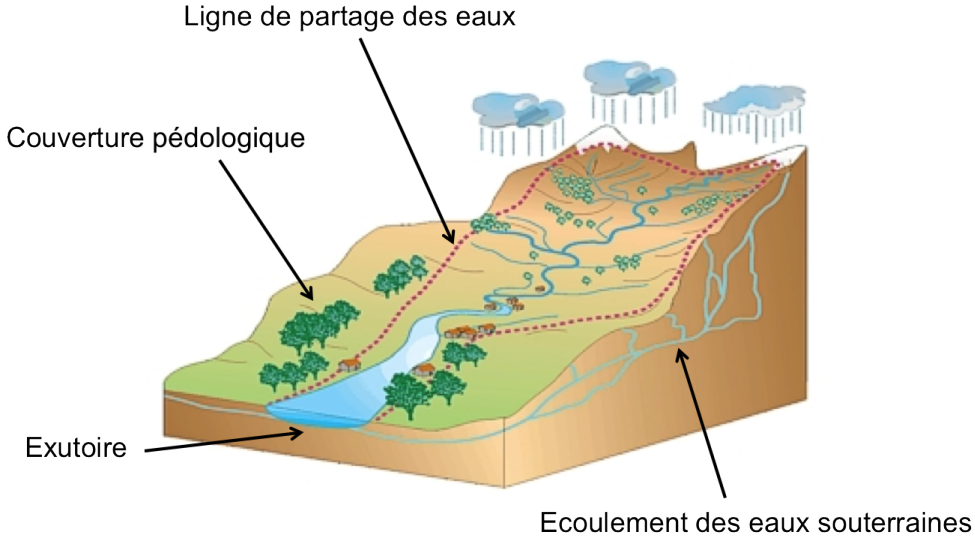

Fig. 1.1 : Schéma d'un bassin versant (d'après le site Web EPTB).

D'après RAMADE (1993), un bassin versant est une zone géographique continentale constituée d'un réseau hydrographique. Il correspond à la totalité de l'aire de capture et de drainage des précipitations. L'aire est symbolisée sur la figure 1.1 par la ligne de partage des eaux.
D'après la synthèse faite par COCHONNEAU *et al.* (2008), l'étude des grands bassins versant est essentielle pour comprendre la dynamique globale de la planète et son évolution future. En effet, leur vaste étendue est sensible à la variation du climat. Ils sont soumis à la forte pression de l'activité humaine (déforestation, activités agricoles et minières, urbanisation) qui a pour conséquence l'altération des sols et de leur couverture végétale. Cette altération provoque à son tour une modification des équilibres hydrologiques et géochimiques allant jusqu'à influencer le régime climatique. Les bassins des grands fleuves tropicaux sont particulièrement étudiés : leurs flux d'eau et de matières drainés contribuent largement au fonctionnement

global de la Terre. Les auteurs BAUMGARTNER et REICHEL (1975) et DEGENS *et al.* (1991) estiment que la contribution de ces bassins est de 57% en d'eau douce, 50% en apports solides, 38% en apports en solution et 45% en carbone organique, par rapport au flux total provenant des continents.
D'après VILLAR *et al.* (2009) le bassin amazonien est le plus vaste de la planète avec environ 6 millions de km^2 : cinq pays se le partagent. Il abrite le premier des fleuves tropicaux dont le débit à l'exutoire est de 209 000 m^3/s et fournit 15% de l'eau fluviale totale.
L'étude de ces grands bassins et le développement d'outils d'analyse et d'interprétation des données acquises sont donc essentiels à la compréhension de leur fonctionnement.

1.2 Mesures scientifiques

Afin de mener à bien l'étude de ces grands bassins, la communauté scientifique s'appuie sur un certain nombre de moyens d'investigation. Nous allons passer rapidement en revue les mesures fluviométriques, pluvio-climatiques et physico-chimiques et les études pédologiques.

1.2.1 Mesures fluviométriques

Mesurées par les stations hydrométriques, elles quantifient les variables hydrologiques telle que la hauteur d'eau moyenne journalière (lecture d'un limnigraphe illustré à la figure 1.2). Cette hauteur d'eau sert à calculer le débit moyen journalier du cours d'eau à l'aide d'une courbe d'étalonnage obtenue par un courantomètre (COCHONNEAU *et al.*, 2006).

1.2.2 Mesures pluvio-climatiques

Elles quantifient les variables météorologiques comme la température, la vitesse du vent et sa direction ainsi que les précipitations moyennes journalières. Elles sont généralement mesurées à l'aide d'une station météorologique.

1.2.3 Mesures physico-chimiques

Effectuées à partir d'échantillons d'eau, elles sont de plusieurs natures (COCHONNEAU *et al.*, 2006). Une partie est effectuée sur le terrain comme la mesure du pH, de la température, de la conductivité électrique et du poids des matières en suspension (par filtration). L'autre partie est effectuée en laboratoire comme la mesure de la concentration des ions majeurs et traces, celle de la silice et du carbone organique dissout et l'analyse du rapport isotopique du strontium (traceur géochimique).

1.2.4 Etudes pédologiques

Elles ont pour but d'établir la distribution spatiale des différentes natures de sol. Appliquées à l'hydrologie, elles sont utiles pour la détermination de la capacité de rétention de l'eau dans le sol d'un bassin versant.

FIG. 1.2 : Photographie de la station du Rio Aiari à Louro Poço (Amazonie).

1.3 Télédétection spatiale et cartographie

La télédétection spatiale apporte une contribution considérable à la cartographie en fournissant des images de la surface de la Terre. D'après ROYER *et al.* (2007), la photographie en relief (stéréoscopie visible) et les Modèles Numériques de Terrain (MNT), le calcul de l'altitude des surfaces continentales et océaniques (altimétrie radar et laser), l'empreinte thermique (lumière infra rouge) ou électromagnétique sont les principales informations données par cet outil. En outre, la répétitivité de l'observation spatiale apporte une dimension temporelle mettant à disposition des séries temporelles de ces mesures.
L'étude de l'émission ou de la réflexion des ondes électromagnétiques à de multiples longueurs d'onde sur différentes surfaces terrestres a permis de les corréler avec les caractéristiques de ces surfaces. L'information spatiale est capable de quantifier la température de surface, l'humidité du sol ou de reconnaître la couverture végétale. Les dernières publications montrent que l'on peut mettre en relation la couleur de l'eau et la concentration en sédiment (MARTINEZ *et al.*, 2007) ou calculer la hauteur de l'eau des fleuves suffisamment larges (ROUX *et al.*, 2008). Le projet de Contribution de l'Altimétrie Spatiale à l'Hydrologie (CASHTEAM, 2006) a démontré qu'il est possible d'utiliser les informations de l'altimétrie spatiale afin de compléter les données hydrologiques issues des réseaux d'observations terrestres, inaugurant la notion de station hydrométrique virtuelle.

1.4 Modélisation mathématique hydrologique

1.4.1 Objectifs

La modélisation mathématique de phénomènes naturels a généralement pour but la description mécanistique du phénomène, la mise en lumière de l'influence de ses facteurs et la prévision de son évolution. D'après MORIN (1991), la modélisation hydrologique est devenue une approche pour résoudre les problèmes de gestion de ressources d'eau (par exemple l'irrigation), d'aménagement de l'environnement (par exemple l'édification de barrage) ou de protection de l'environnement (évolution due à la pression anthropique) et de prévision de catastrophes naturelles (crues). Les modèles hydrologiques se présentent sous la forme de formules mathématiques et d'algorithmes. Ils sont déterministes quand ils ne contiennent aucun élément aléatoire mais sont le plus souvent paramétriques afin de rendre compte de variables aléatoires. En général, les modèles hydrologiques se basent sur les séries de mesures temporelles issues des observations décrites ci-dessus.

1.4.2 Musckingum-Cunge

L'un de ces modèles, Musckingum-Cunge (CUNGE, 1969) qui est intégré à la plateforme Umodelis, calcule le débit d'un tronçon de cours d'eau (bief) par propagation. Il consiste en un ensemble d'équations différentielles qui mettent en jeu les caractéristiques du bief (longueur, pente, largueur) et le débit du bief précédent.

1.5 Contribution informatique

L'étude des bassins versants utilise massivement l'outil informatique. On distingue trois contributions majeures : la présentation et le traitement de l'information notamment avec les Systèmes d'Information Géo-

graphique (SIG), le stockage de l'information avec les bases de données et les serveurs de cartes et le calcul pour l'exécution de simulations de modèles.

1.5.1 SIG

Les SIG ont pour but la représentation graphique des informations géoréférencées issues principalement de la télédétection et de la cartographie. Cette représentation prend la forme d'un assemblage de couches d'information de même nature. Les outils intégrés dans le SIG apportent l'analyse ou le traitement de l'information spatialisée.
Les données manipulées par les SIG sont principalement de deux natures : les images matricielles géoréférencées appelées rasters (grids) formées d'une matrice de pixels auxquels sont affectées une ou plusieurs valeurs ou les images vectorielles formées à partir d'objets géométriques basiques (lignes, points, polygones, etc.). La figure 1.3 illustre le concept d'organisation de l'information géographique sous forme de couches de rasters et d'images vectorielles. Les couches sont contenues dans une ou plusieurs cartes (maps). L'orientation objet des images vectorielles permet l'introduction de propriétés d'objets pouvant être des mesures scientifiques.

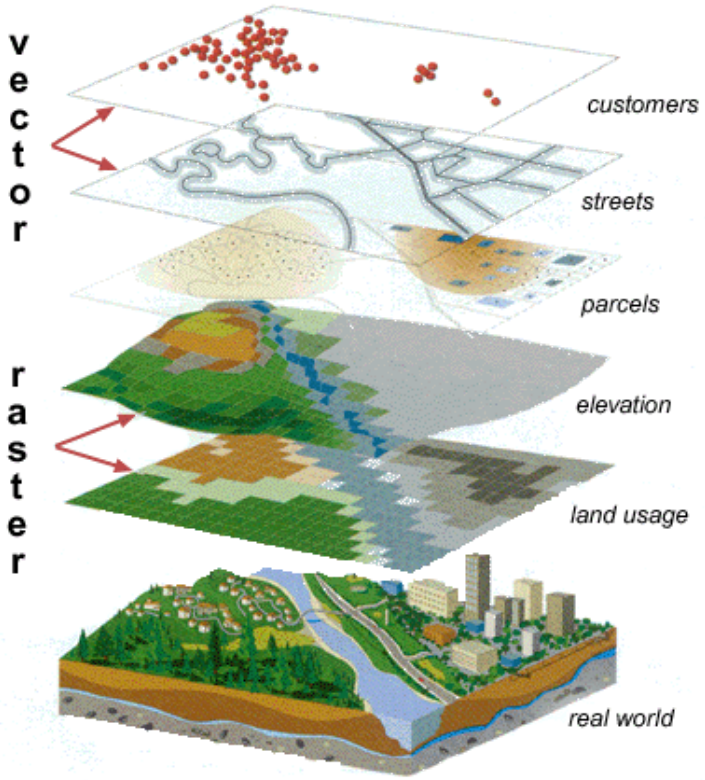

Fig. 1.3 : Illustration du concept de représentation en couches (layers) dans un SIG (d'après le site Web department of geomatics, Melbourne university).

1.5.2 Bases de données et serveurs de cartes

Les bases de données sont le moyen classique de stockage numérique et de distribution des données obtenues par les différentes mesures. A l'image des bases de donnés, les serveurs de cartes organisent le stockage et la répartition des cartes et des couches créées par les SIG. A noter l'apparition d'extension spatiale aux bases de données afin d'introduire la notion de géoréférencement des données.

1.5.3 Calcul

L'informatique apporte la puissance de calcul nécessaire à la résolution des équations mathématiques de la modélisation hydrologique. Les simulations sont des exécutions de modèles mathématiques selon un ensemble de paramètres donnés.

1.6 Problématique

Les précédentes sections ont présenté les principaux outils à disposition pour l'étude des bassins versants. Cette section montre la complémentarité de ces outils puis souligne le besoin d'une plateforme fédératrice et ouverte. Enfin, elle présente l'intérêt de la comparaison des outils et de les mettre les uns au service des autres.

1.6.1 Complémentarité

Le processus d'étude des bassins versants amène le chercheur à exploiter la complémentarité des outils présentés ci-dessus. Pour ne prendre que l'exemple de l'hydrologie, les mesures sur le terrain permettent de calibrer les modèles hydrologiques. Lorsqu'il n'est pas possible de maintenir des stations hydrométriques ou lorsque l'étendue du bassin est trop vaste comme c'est le cas du bassin amazonien, la télédétection apporte un recours pour compléter les données du terrain (ROUX *et al.*, 2008) avec les stations hydrométriques virtuelles et la surveillance pédologique. La cartographie alimentée par la télédétection et gérée par les SIG permet de géolocaliser les phénomènes naturels étudiés (flux d'eau et de matière).

1.6.2 Besoin d'une plateforme fédératrice et ouverte

Pour COCHONNEAU *et al.* (2008) les travaux scientifiques supposent une analyse des données décrites ci-dessus et font intervenir des communautés scientifiques différentes (climatologues, hydrologues, pédologues, géochimistes, géodynamiciens etc.) disposant chacun de leurs outils et méthodes propres. Mais de nombreuses données et méthodes d'analyse sont communes. La complémentarité des techniques et les données en commun entre ces disciplines ont fait naître le besoin d'unifier les outils les exploitant au sein d'une même plateforme. L'aspect pratique de la plateforme, la simplicité de son accès sont mis en avant ainsi que l'abstraction des problèmes informatiques aux chercheurs non informaticiens ou encore l'utilisation de la plateforme comme outil de formation pour les étudiants. Enfin, la plateforme a un rôle d'intégration des travaux scientifiques (outils de traitements et modèles hydrologiques) dans un environnement d'exploitation opérationnel.

1.6.3 Comparaison des outils

Pouvoir comparer les résultats issus de différents modèles mathématiques traitant du même problème apporte une aide au choix de l'outil le mieux adapté, par exemple afin de résoudre un problème de changement d'échelle pour les modèles de pluie-débit (modèle quantifiant le lien entre les précipitations et le débit d'un bassin à son exutoire).

1.6.4 Chaînage d'outils

L'idée de chaîner les outils et les modèles apparaît pertinente afin d'exploiter leur complémentarité. Les résultats de l'un sont les entrées du suivant. Chaque outil étant expert dans son domaine de validité, leur collaboration permet de décrire des systèmes plus complexes, plus proches de la réalité.

La communauté scientifique pose un problème d'ingénierie logicielle afin de valoriser ses travaux et de simplifier son utilisation. Le chapitre suivant aborde la demande précise en développement logiciel.

1.7 Instituts participant

Ce mémoire décrit les phases de conception et de développement du projet Umodelis pour une plateforme de modélisation hydrologique. Le projet Umodelis a été financé par L'Institut de Recherche pour le Développement (IRD) dans le cadre des programmes SPIRALES sous la responsabilité de Madame Marie-Paule Bonnet, Chargé de Recherche IRD et la direction de Monsieur Gérard Cochonneau, Ingénieur de Recherche IRD, membres de l'équipe Eau Sol Environnement (ESE) du Laboratoire des Mécanismes et Transferts en Géologie (LMTG) sous la tutelle de L'IRD, du Centre National de la Recherche Scientifique (CNRS) et de l'Université Paul Sabatier (UPS), laboratoire appartenant au campus de l'Observatoire Midi Pyrénées (OMP) à Toulouse. Le mémoire s'est déroulé de juillet 2007 à juillet 2008 à Brasilia avec le concours de la délégation IRD Brésil, en collaboration et au sein de l'Agência Nacional de Águas (ANA ; l'agence nationale de l'eau). Les données hydro-climatiques et cartographiques sont issues de L'Observatoire de Recherche en Environnement contrôles géodynamique, HYdrologique et biogéochimique de l'érosion/altération et des transferts de matières dans le bassin de l'AMazone" (ORE-HYBAM), qui est un des projets de l'IRD au Brésil. Ce qui suit est une brève description des instituts qui ont permis ce mémoire.

1.7.1 IRD et IRD Brésil

L'IRD est un Etablissement Public à caractère Scientifique et Technologique (EPST) français qui a pour mission de mener des programmes de recherche en partenariat avec les instituts des pays du Sud et assure des formations (doctorats, masters). Ses principales thématiques sont la lutte contre la pauvreté, l'étude des migrations internationales, des maladies émergentes infectieuses, du changement climatique, des ressources en eau et des écosystèmes et ressources naturelles. L'IRD rassemble 800 chercheurs et 1000 ingénieurs et intervient dans plus de cinquante pays (IRD, 2007).
L'IRD Brésil est la représentation de l'IRD dans le plus grand pays de l'Amérique du Sud. La très grande diversité de ce pays en fait un terrain privilégié de recherches scientifiques (IRDBRESIL, 2006). En collaboration avec le Conselho Nacional de Desenvolvimento Cientifico e Tecnológico (CNPq ; conseil national du développement scientifique et technique), principal partenaire, mais aussi avec d'autres partenaires privés

ou publics comme les universités, l'IRD Brésil mène des projets touchant les milieux physiques continentaux et les influences océaniques, la gestion des ressources naturelles et de l'environnement, la santé et l'environnement, les identités, territoires et développement en Amazonie, la société, la ville et l'économie. Elle représente en moyenne 35 agents expatriés.

1.7.2 ORE-HYBAM

Le réseau d'observations ORE-HYBAM est né de la volonté d'acquisition de données in-situ, satellitaires et géographiques du bassin amazonien (GUYOT, 2005 ; COCHONNEAU *et al.*, 2006) telles qu'elles ont été décrites ci-dessus. Depuis 2003, des données journalières sont mises à disposition au moyen de bases de données réparties consultables, entre autres, à l'aide du site Web de l'observatoire.

1.7.3 LMTG

Le LMTG et en particulier l'équipe ESE a pour mission d'étudier le transfert d'eau et des éléments associés (matières en suspension, éléments majeurs et traces) à la surface des continents et à l'interface avec l'océan. Elle a également l'ambition de développer les pratiques et usage des eaux et leurs impacts sur les plans économique et juridique.

1.7.4 ANA

L'agence nationale de l'eau brésilienne a pour but la gestion des ressources hydriques, la régulation de leurs utilisations et la promotion d'une utilisation raisonnée. L'ANA est le principal partenaire pour toutes les études hydrologiques des bassins versants du Brésil. Elle a développé plusieurs outils dont certains en collaboration avec l'IRD.

1.8 Guide de lecture du mémoire

Au cours des chapitres, nous aborderons la demande et les contraintes appliquées au développement du projet Umodelis, quelques éléments de géomatique pour la compréhension du reste du mémoire et la description du projet Umodelis avec un état de l'art des plateformes de modélisation. Les chapitres suivants décriront l'architecture de la plateforme Umodelis, expliqueront le choix du SIG User-friendly Desktop Internet GIS (uDig) et la stratégie de son intégration dans la plateforme. Les derniers chapitres auront pour sujet la conception de la bibliothèque Umodelis, celle du module de création de sites hydrologiques, celle du module d'exploitation de résultats et enfin la conception du serveur de modélisation hydrologique.
Conformément aux prescriptions du CNAM les termes techniques étrangers ne sont pas utilisés s'il en existe une traduction (le terme étranger est donné entre parenthèses). Le dictionnaire de terminologie québécois est pris comme référence. Cependant, si la traduction est inexistante ou insatisfaisante (non intuitive à cause de l'emploi systématique du terme étranger), le terme étranger est utilisé.

CHAPITRE 2

Demande et contraintes

Ce chapitre décrit la demande de développement d'une plateforme de modélisation hydrologique formulée par la communauté scientifique et également les contraintes imposées à sa réalisation.

2.1 Demande

Les chercheurs et ingénieurs de l'IRD et de leurs partenaires souhaitent une plateforme représentant un socle sur lequel pourront se fixer les membres des quatre composantes de la modélisation des bassins versants. Ces composantes seront décrites dans les sections suivantes ainsi que le besoin de répartition des ressources de la plateforme et les idées directrices pour le développement de son Interface Homme Machine (IHM). Dans un premier temps, la communauté scientifique attend une base présentant les propriétés demandées mais qui soit néanmoins fonctionnelle concernant la modélisation hydrologique.

2.1.1 Fédération des quatre composantes

Comme le chapitre introductif l'a montré, l'étude des grands bassins versants nécessite l'exploitation de plusieurs types d'outils et de données. Ces outils et données peuvent être classés sous quatre composantes que la plateforme doit intégrer. La plateforme ne représente pas seulement un point d'accès à ces ressources mais aussi une base où pourront s'agréger les futurs travaux. Les quatre composantes sont :

2.1.1.1 Bases de données

Les bases de données satellitaires d'altimétrie radar et imagerie optique (MNT) ainsi que les bases de données d'observations in situ hydro-climatologiques et physico-chimiques doivent être accessible via la plateforme.

2.1.1.2 Outils de traitements de données

La plateforme doit faire le lien avec les outils de traitements développés antérieurement en fournissant un patron spécifiant leur intégration. Par exemple, le logiciel Hydraccess (VAUCHEL, 2004) qui offre une interface pour le stockage et les traitements de données hydrologiques de l'ORE-HYBAM, est attendu.

2.1.1.3 Modèles mathématiques

La plateforme doit donner un patron et une architecture suffisamment génériques pour intégrer les différentes formes de modèles mathématiques. L'accès aux modèles ainsi que leur contrôle sont plus précisément décrits dans la section 2.1.2 page 10.

2.1.1.4 SIG et spatialisation des données

Un SIG doit être intégré à la plateforme afin de visualiser et de manipuler les données cartographiques mais aussi de permettre la spatialisation des données hydrologiques. L'idée est d'utiliser un SIG pour présenter ces données afin de les transformer au format d'entrée des modèles.

2.1.2 Répartition des ressources

Les points précédents expriment le souhait d'une centralisation des données et des outils de traitements pour l'étude des bassins versants. Cette section présente le besoin d'un accès réparti aux modèles mathématiques, aux données générées par la plateforme et aux moyens de calcul. La communauté scientifique souhaite accéder aux modèles mathématiques et aux données associées depuis n'importe quel poste de travail et sous entend l'exécution et le contrôle de simulations de modèles à distance afin de profiter des ressources des centres de calcul (par exemple la ferme de calcul du LMTG). Ce dernier souhait est important pour les équipes ne possédant pas de fortes ressources de calcul. En revanche, il n'est pas demandé de répartir l'accès aux outils de traitements de données car ils ne sont pas nécessairement très grands consommateurs de ressources de calcul. Néanmoins, l'architecture d'Umodelis doit prévoir une solution pour d'éventuelles exceptions.

2.1.3 Interface Homme Machine

Elle doit fournir les interfaces afin de paramétrer les outils et les modèles intégrés à la plateforme et permettre l'affichage et l'édition des données et résultats de simulations, soit par la biais du SIG concernant les données cartographiques, soit par ses propres interfaces pour les autres types de données (données non spatialisées). Elle doit offrir également le moyen de superviser l'exécution des outils et surtout de contrôler les simulations de modèles. Elle restituera les résultats de simulations sous forme de rapports, de graphiques ou éventuellement d'objets en base de données. Différentes versions de l'IHM sont attendues, une version de type client riche (ou lourd) intégrant ou intégré à un SIG et une version de type client léger fonctionnant au travers d'un navigateur Web et dépourvu de SIG.

2.2 Contraintes de développement

Cette section présente les contraintes imposées au projet Umodelis dues à son environnement et à ses utilisateurs. Les contraintes portent sur les données traitées, sur l'adaptation du projet à son environnement ainsi que sur l'intégration des outils existants et sur le développement informatique du projet.

2.2.1 Hétérogénéité des données

Les données manipulées par Umodelis sont de différentes natures comme nous l'avons vu précédemment. Elles peuvent être graphiques avec les objets géographiques, alphanumériques spatialisées ou non spatialisées. De plus, elle proviennent de différentes sources comme les serveurs de cartes, les systèmes de fichiers ou les bases de données. Enfin, leurs métadonnées ne sont pas normalisées et vu la masse d'information, il n'est pas envisageable de le faire. La plateforme Umodelis doit donc masquer l'origine des données aux

utilisateurs, s'adapter aux différentes métadonnées et proposer des structures de données utilisables par tous les outils de traitements.

2.2.2 Adaptation à l'existant

Umodelis s'inscrit dans un environnement hétérogène. Les bases de données de l'IRD fonctionnent sous les Systèmes de Gestion de Bases de Données Relationnels (SGBDR) MySQL (Sun Microsystems ; licence General Public License ou GPL) et PostgreSQL (licence Berkeley Software Distribution ou BSD) avec l'extension spatiale PostGIS. Les modèles hydrologiques sont essentiellement écrits en Fortran (plusieurs standards et compilateurs utilisés) et quelques modèles le sont en Java (GPL ; Sun Microsystems). Enfin, les outils autonomes comme Hydraccess (licence IRD) doivent être à terme intégrés à la plateforme. La plateforme doit être fédératrice des outils et modèles existants et à venir.

2.2.3 Développement

Les principaux utilisateurs de la plateforme étant les chercheurs et les étudiants de l'IRD et de ses partenaires, l'IRD souhaite que la plateforme soit distribuée sous licence GPL voire Lesser General Public License (LGPL). Ces licences impliquent que les composants sur lesquels s'appuie Umodelis doivent être distribués sous des licences compatibles. Il en va de même pour les outils de développement de la plateforme afin d'assurer sa maintenance.
Les éventuels composants de la plateforme hébergés sur les machines des utilisateurs doivent fonctionner sous les principaux systèmes d'exploitation : Windows XP (Microsoft) et supérieur, Linux (noyau 2.6) et Mac OS X (Apple) 10.4 et supérieur.
A des fins de pérennité, la plateforme doit être la plus indépendante possible vis à vis du SIG afin de minimiser l'impact de son éventuel remplacement.
L'équipe de développement doit éviter la réécriture des modèles hydrologiques pour leur intégration dans la plateforme. Au pire, il sera toléré une réorganisation du code.
Enfin, il est recommandé qu'Umodelis respecte les standards émergeant promus par l'Open Geospatial Consortium (OGC), tendance de la géomatique actuelle, dans le but d'assurer la compatibilité et la pérennisation d'Umodelis.

2.3 Conclusion

Les scientifiques de l'IRD et de ses partenaires attendent les fondations d'une plateforme dont le but est de mutualiser les traitements et les données nécessaires à l'étude des bassins versants. Dans un contexte où les modèles mathématiques et les données sont répartis, Umodelis en représente un point d'accès. Elle comportera un SIG afin de faciliter son utilisation et concernant les contraintes, elle devra à la fois s'intégrer dans un environnement hétérogène et minimiser ses dépendances extérieures. Le chapitre suivant est consacré à quelques éléments de géomatique afin d'améliorer la compréhension de ce document.

CHAPITRE 3

Éléments de géomatique

Ce chapitre présente quelques éléments de géomatique afin de préciser les concepts utilisés et d'introduire des notions propres à la plateforme Umodelis comme le site hydrologique et le projet. Il commence par la description des objets géographiques manipulés, continue avec la définition de la donnée spatialisée, puis celle du site hydrologique, et finit par la notion de projet.

3.1 Objets géographiques

Les objets géographiques représentent graphiquement des concepts géographiques au sein d'images vectorielles. Ils s'accompagnent éventuellement de propriétés (attributs). L'orienté objet pour les images rasters, basé sur la reconnaissance de forme, ne sera pas abordé dans ce mémoire.

3.1.1 Réseau hydrographique (hydrographic network)

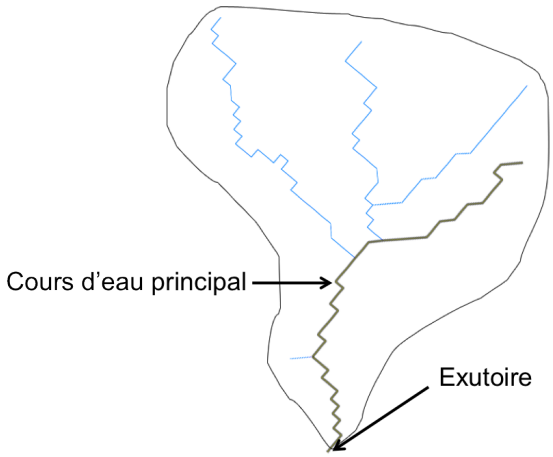

FIG. 3.1 : Modélisation d'un réseau hydrographique d'un bassin versant.

Un réseau hydrographique est constitué de cours d'eau (stream). Un des cours d'eau est qualifié de principal (sélectionné sur la figure 3.1) car il conduit à l'exutoire du bassin (dont la limite de partage des eaux est représentée par l'enveloppe noire sur la figure 3.1), les autres cours d'eau sont ses affluents. Les cours d'eau sont souvent représentés par un ensemble de points liés par interpolation linéaire (line string). Ils sont généralement représentés en deux dimensions ce qui simplifie leur modélisation (ils n'ont pas de largeur). Les propriétés des cours d'eau sont nombreuses, on peut retenir les plus importantes comme leur nom, leur longueur, la pente moyenne entre les deux extrémités, leur profondeur moyenne ainsi que leur vitesse journalière d'écoulement.

3.1.2 Station hydrométrique (hydrometric station)

Elle est généralement représentée par un point situé aux coordonnées de la station. Le nom, les coordonnées, le type (in situ, virtuelle dans le cas de l'altimétrie satellitaire), la surface drainée et le nivellement font partie des propriétés des stations.

3.1.3 Tributaire ou affluent (tributary)

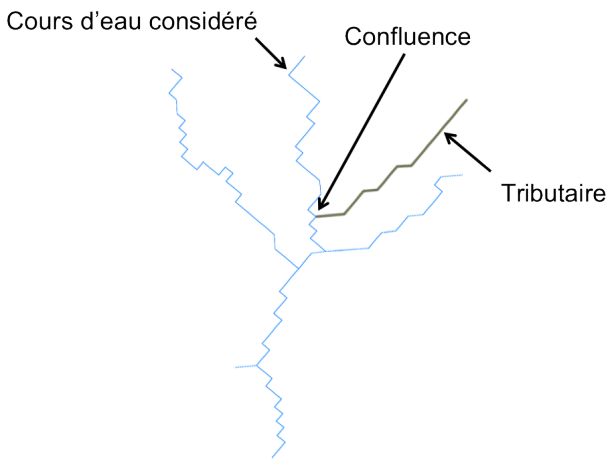

FIG. 3.2 : Illustration des notions de tributaire et de confluence.

Si l'on considère un cours d'eau en particulier, un tributaire est un cours d'eau de plus petit débit qui se jette dans celui-ci en un point appelé confluence. Si l'on veut représenter uniquement le cours d'eau principal, le tributaire peut être réduit à son point de confluence et à la valeur de son débit à ce point ci. Le tributaire prend alors la représentation d'un point aux coordonnées de la confluence, pondéré du débit du tributaire à la confluence. Si l'on reprend la figure 3.2, on s'aperçoit que le cours d'eau sélectionné est un tributaire du cours d'eau considéré qui est lui même tributaire du cours d'eau principal du bassin (sélectionné sur la figure 3.2). La notion de tributaire est une notion imbriquable.

3.1.4 Bief (reach)

Un bief est une portion de cours d'eau définie entre deux objets géographiques. Ces objets peuvent être, par exemple, deux confluences (figure 3.3) ou deux stations hydrométriques (figure 3.4). Le bief est utilisé comme l'unité des cours d'eau dans les modèles hydrologiques linéaires.

3.2 Données spatialisées

Les données spatialisées sont des données géographiquement localisables. Elles peuvent être sous forme de propriétés (attributs) des objets géographiques ou de données raster (les raster et les images vectorielles sont géoréférencées).

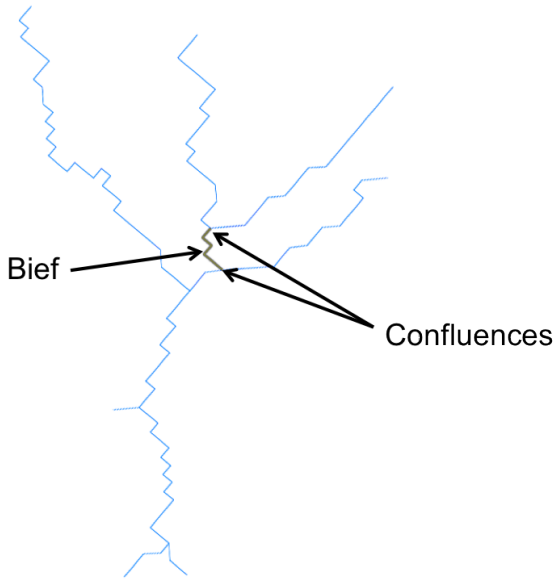

FIG. 3.3 : Bief défini entre deux confluences.

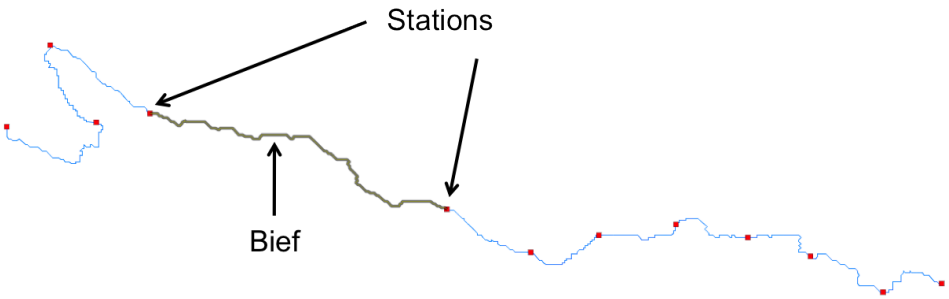

FIG. 3.4 : Bief défini entre deux stations hydrométriques.

3.3 Sites hydrologiques

Notion apportée par Umodelis, les sites hydrologiques sont des objets géographiques, comme un cours d'eau, auquel l'utilisateur peut leur associer d'autres objets géographiques, comme un tributaire ou des données hydrologiques spatialisées, par exemple une aire de drainage ou des données hydrologiques non spatialisées comme la longueur du cours d'eau. Les sites sont représentés graphiquement par l'ensemble de ses objets géographiques et alpha numériquement par ses données spatialisées ou non. Les données spatialisées associées aux sites hydrologiques sont extraites de rasters ou sont calculées à partir des propriétés des objets géographiques mais sont indépendantes de celles-ci. En effet, la notion de sites hydrologiques étend celle de propriété d'objets géographiques en intégrant d'autres objets géographiques et des données non spatialisées. Enfin, les propriétés des objets géographiques associés aux sites ne sont pas retenues afin d'éviter une redondance de l'information. La figure 3.5 schématise l'ensemble de ces notions.

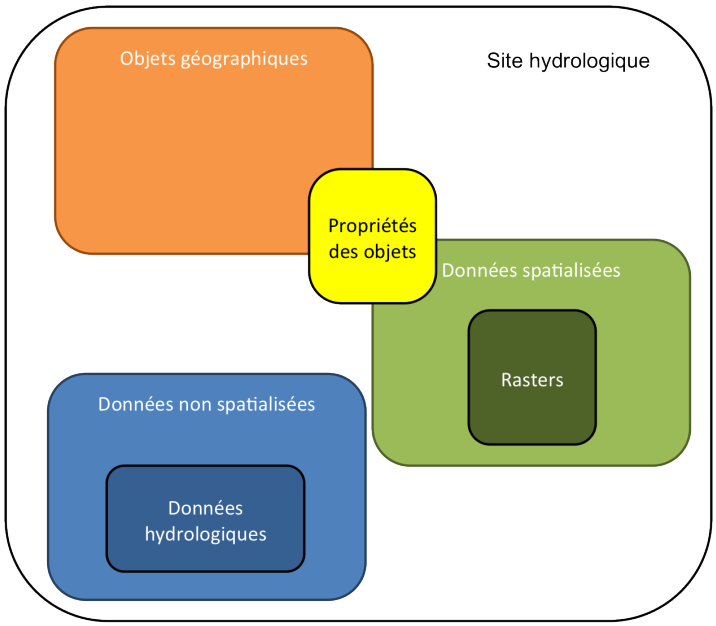

Fig. 3.5 : Schéma illustrant la relation entre les différents concepts présentés dans le chapitre.

3.4 Projet (project)

Cette notion n'est pas à proprement parler un site hydrologique mais est un ensemble cohérent (au sens hydrologique) de sites hydrologiques et comporte quelques paramètres utiles aux modèles hydrologiques.

3.5 Conclusion

Réseau hydrographique, station hydrométrique, bief et tributaire ne sont que quelques uns des concepts géomatiques que la plateforme manipule actuellement. Nous verrons au chapitre suivant un état de l'art

sur des plateformes de modélisation hydrologiques analogues à Umodelis afin de présenter l'originalité du projet.

CHAPITRE 4

Projet Umodelis

Les précédents chapitres ont placé le projet Umodelis dans son contexte scientifique et ont explicité la problématique liée à son contexte. Ce chapitre se propose de présenter les objectifs du projet en réponse à une partie des besoins précédemment exprimés. Dans un premier temps, il sera fait un état de l'art concernant les plateformes de modélisation hydrologique. Dans un second temps, les objectifs du projet Umodelis seront présentés selon les axes suivants : conception d'une architecture ouverte, modules de traitements et de modèles hydrologiques.

4.1 Etat de l'art des plateformes de modélisation hydrologique

Selon YINHUAN et QIUMING (2007), l'idée de spatialiser les modèles hydrologiques a déjà été implémentée dans plusieurs projets informatiques. Le but de cet état de l'art est de comparer les différentes approches qui ont été développées, avec la problématique et les contraintes précédemment données. Il existe jusqu'à ce jour trois projets : l'ATelier HYdrologique Spatialisé (ATHIS), OpenFluid et LIQUID qui ont en commun, avec Umodelis, d'être des environnements de modélisation hydrologique et de traitements de données spatialisées couplés à un SIG. Cet état de l'art ne traitera pas des plateformes de modélisation généralistes comme The Invisible Modelling Environment (TIME ; RAHMAN *et al.*, 2003), Dynamic Information Architecture System (DIAS ; CHRISTIANSEN, 2000) et Spatial Modelling Environment (SME ; VOINOV *et al.*, 1999) ni des standards ou architectures pour le développement de plateformes comme Open Modelling Interface (OpenMI ; GREGERSEN *et al.*, 2005) et Jena Adaptable Modelling System (JAMS ; KRALISCH et KRAUSE, 2006).

4.1.1 ATHYS

Présenté pour la première fois par BOUVIER *et al.* (1996) et maintenu par le laboratoire Hydrosciences (IRD/CNRS, Montpellier), ATHYS est une plateforme composée de quatre modules (figure 4.1). Le module MERCEDES, qui est chargé de la modélisation hydrologique spatialisée, intègre plusieurs fonctions hydrologiques dont la prévision des crues, la gestion de la ressource en eau et les études d'impact liées aux changements géographiques ou anthropiques. L'architecture du module est ouverte : il est possible d'y intégrer de nouvelles fonctions (faculté non documentée à ce jour). Le module VISHYR offre aux utilisateurs des fonctions de visualisation et de traitement de données hydro-climatiques pré et post simulation. Enfin, le module VICAIR est chargé de la visualisation et du traitement des données géographiques spatialisées et le module SPATIAL de l'interpolation spatiale.
ATHIS est un environnement opérationnel très riche en fonctionnalités. Ces modules de traitements de données hydro-climatiques et de modélisation hydrologique sont des atouts majeurs pour la compréhension des processus naturels.

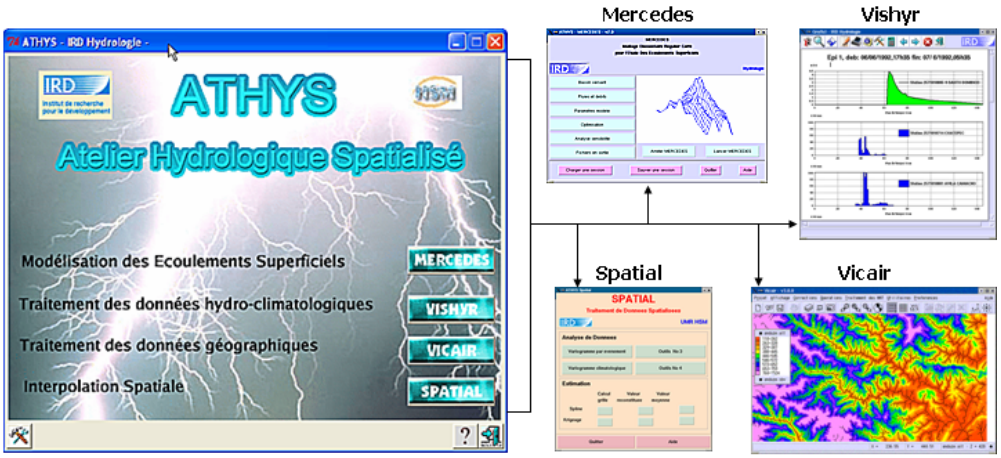

FIG. 4.1 : Capture d'écran de la plateforme ATHYS (d'après le site Web ATHYS).

Néanmoins, comme le soulignent les auteurs, le SIG d'ATHYS est limité : le module VICAIR est uniquement orienté imagerie. S'il permet d'exploiter la plupart des sources d'informations de type raster (MNT, photos, images, etc.), il ne supporte pas la représentation d'images vectorielles (objets géographiques). L'unique concept utilisé est le pixel. Ce module est inadapté à la tendance de la géomatique actuelle : la représentation géographique orientée objet.
D'autre part, ATHYS n'est pas un environnement distribué, il n'utilise que les systèmes de fichiers. Même si ces fichiers sont standardisés ou s'il existe des fonctions d'exportation vers d'autres formats standard, fondamentalement ATHYS est un environnement monoposte isolé. Il n'est pas conçu pour le travail en collaboration, la distribution des calculs, des données et des traitements. Enfin, la documentation sur l'intégration de nouvelles fonctions n'étant pas disponible, il n'a pas été possible d'apprécier l'extensibilité d'ATHYS.

4.1.2 OpenFLUID

Historiquement issu du projet MHYDAS (MOUSSA *et al.*, 2002), un modèle mathématique pluie-débit, OpenFLUID se présente sous la forme d'un moteur de modélisation (OpenFLUID-Engine), d'une interface graphique (OpenFLUID-Builder) et d'un site web communautaire (OpenFLUID-Web), comme l'illustre la figure 4.2.
Développée par le Laboratoire d'étude des Interactions Sol - Agrosystème – Hydrosytème (LISAH) de Montpellier, l'originalité de cette plateforme est de proposer la création de modèles hydrologiques à partir de l'assemblage de fonctions hydrologiques (figure 4.3). Les modèles hydrologiques gagnent ainsi en flexibilité.
Etant conçue pour la modélisation de petits bassins versant (moins de 100 km^2), elle propose un fin découpage spatial de la nature des sols afin d'étudier spécialement les bassins fortement anthropisés (aménagements agricoles) et des fonctions hydrologiques adaptées à cette échelle.
L'utilisateur effectue les traitements pré et post simulations par le biais d'OpenFLUID-Builder tandis que la partie simulation est assurée par OpenFLUID-Engine. L'extension des fonctions hydrologiques est possible grâce à la mise à disposition d'un environnement de développement (Eclipse C++), d'une documentation

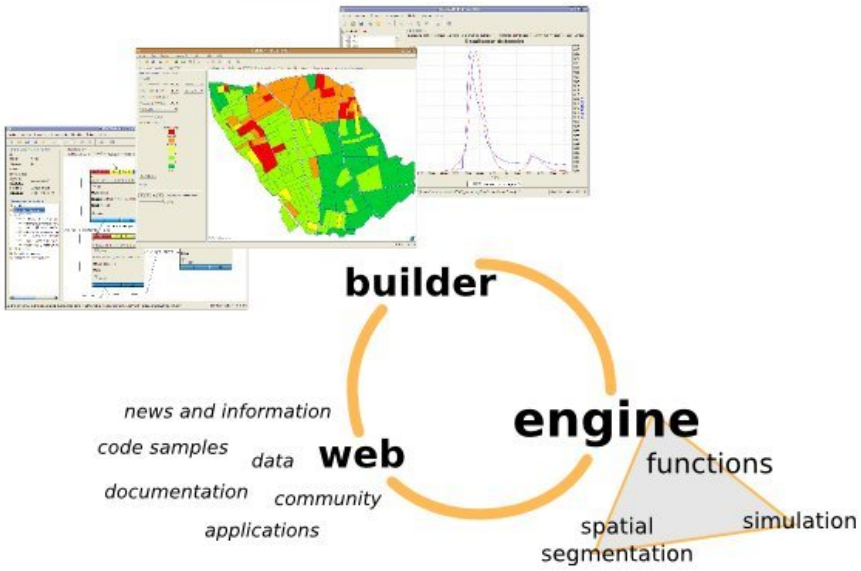

FIG. 4.2 : Composants de la plateforme OpenFLUID (d'après le site Web OpenFLUID).

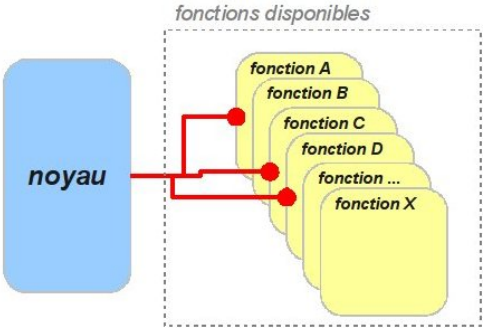

FIG. 4.3 : Exemple de création d'un modèle par assemblage de fonctions hydrologiques (d'après le site Web OpenFLUID).

et l'aide d'une communauté d'utilisateurs et de développeurs.
OpenFLUID-Builder est un SIG implémenté par l'équipe de développement utilisant les composants GeoTools, bien que complet pour l'utilisation conjointe d'OpenFLUID-Engine, les données et les modèles ne sont pas dépendants. Les deux modules étant autonomes et communiquant via des fichiers standardisés en eXtensible Marckup Language (XML), rien n'interdit l'utilisation d'un SIG tiers pour la génération automatique des fichiers d'entrée des modèles.
Du point de vue scientifique, cette plateforme est dédiée à l'étude des petits bassins agricoles. Les échelles de temps et d'espace concernées ne sont donc pas du tout adaptées à la modélisation des grands bassins. Enfin, son fondement est également différent, puisqu'il ne s'agit pas d'une plateforme de capitalisation des outils et modèles déjà existants, mais plutôt, d'une plateforme proposant différentes formulations selon l'échelle spatiale étudiée.

4.1.3 LIQUID

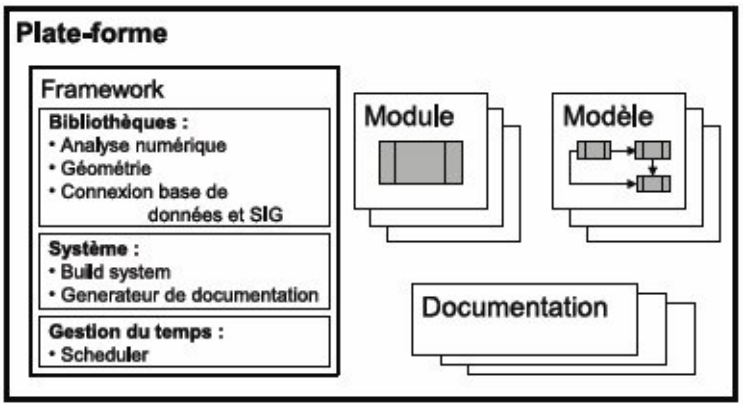

FIG. 4.4 : Composants de la plateforme LIQUID (d'après BRANGER, 2007).

LIQUID (VIALLET *et al.*, 2006) est une plateforme proposée par la société d'hydro-informatique HYDROWIDE en partenariat avec le Laboratoire d'étude des Transferts en Hydrologie et Environnement de Grenoble (LTHE) et le Cemagref. Plateforme modulaire (figure 4.4), elle comporte une bibliothèque de processus hydrologiques qui servent, à l'instar d'OpenFLUID, à assembler des modèles dont quelques uns sont proposés. Elle comporte également un moteur de réalisation de simulations, un système de génération de documentations basé sur l'assemblage des processus hydrologiques et un SIG comme interface graphique. Elle est orientée bases de données et son extensibilité repose sur des patrons de conception de modules.
Un module, au sens de cette plateforme, est composé d'un schéma de données, d'un préprocesseur de données et d'un solveur (figure 4.5). Le schéma de données décrit l'ensemble des données, leur organisation et leur format. Le schéma de données est intimement lié au modèle intégré dans le module. En effet, il renseigne le préprocesseur sur la façon d'extraire les données (1) depuis les bases de données de l'utilisateur (2) et les mettre en forme (3) afin qu'elles puissent être utilisées par le solveur (4). Cette approche permet, notamment, de résoudre les problèmes d'hétérogénéité des métadonnées de bases de données. Le solveur est le cœur du module : il résout les équations différentielles du modèle hydrologique implémenté par pas de temps qui est géré par l'ordonnanceur de la plateforme (scheduler sur la figure 4.4).

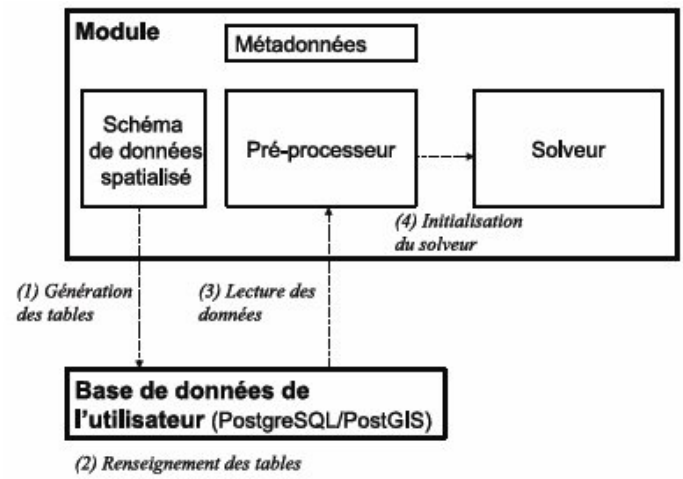

FIG. 4.5 : Architecture d'un module au sein de LIQUID (d'après BRANGER, 2007).

Bien que proche des besoins exprimés précédemment, LIQUID privilégie le développement des modules spécialement conçus pour elle et non l'intégration de modèles existants. Elle est développée sous Windows en C++ (STL et Boost) mais elle serait facilement portable sous un autre système d'exploitation. LIQUID constitue une approche convergente avec Umodelis. Sa conception laisse à penser qu'il est possible de développer des modules se connectant à des serveurs de ressources géomatiques, ajoutant ainsi un autre tiers à l'application. Des contacts existent au sein du LTHE et une collaboration avec HYDROWIDE est envisagée.

4.1.4 Synthèse

Ces plateformes, bien que riches en fonctionnalités scientifiques et dont la structure nous a inspiré, n'apportent pas une solution totalement satisfaisante à la problématique posée. Le principal manque est l'absence d'architecture capable d'intégrer d'autres applications. L'aspect distribué des traitements n'est pas implémenté. Le parti pris de ces plateformes pour le développement interne des modèles hydrologiques n'apporte pas la flexibilité nécessaire à l'intégration des outils existants ainsi que la souplesse pour l'intégration de nouveaux travaux de recherche. Enfin, le bénéfice de la puissance, l'évolutivité et la pérennité d'un SIG à part entière, indépendant de la plateforme mais intégré à celle-ci constitue un argument décisif pour le développement d'une nouvelle plateforme de modélisation hydrologique.
La communauté souhaite donc pouvoir disposer d'une application qui permet de construire des modèles sur-mesure, en combinant souplesse, évolutivité et rapidité de mise en œuvre. Une architecture résolument ouverte aux applications, dans un contexte distribué. C'est dans cette optique que le projet Umodelis a été développé.

4.2 Objectifs du projet Umodelis

Dans un premier temps, l'équipe du projet, consciente des moyens humains et financiers réduits, s'est concentrée sur la conception et le développement d'une solution appliquée à une partie de la problématique. Cette base de la plateforme permet un développement futur de toutes les fonctionnalités attendues par la

communauté scientifique. Ainsi l'équipe a dégagé les objectifs du projet Umodelis : livrer une plateforme de modélisation dont les caractéristiques sont les suivantes :

4.2.1 Architecture ouverte

Umodelis est une plateforme paramétrable capable d'intégrer les travaux de recherche déjà existants et ceux à venir. Couplée avec un client SIG ou accessible à l'aide d'un client Web, elle permet la mise en forme des données en entrée des modèles hydrologiques, le contrôle à distance des simulations de ces modèles, met à disposition des ressources en calcul numérique puissantes et est accessible depuis n'importe quel poste de travail. Elle respecte les contraintes indiquées au chapitre 2 page 9.

4.2.2 Modules de traitements

4.2.2.1 Création de sites d'études

En termes de fonctionnalité, le projet Umodelis prévoit un outil de création de sites d'études hydrologiques ou sites hydrologiques dont la conception et l'implémentation m'ont été confiées. Cet outil consiste en l'assemblage de données spatialisées ou non et d'objets géographiques, provenant de plusieurs sources et de métadonnées hétérogènes. Les données sont alors représentées, selon leur nature, par une structure de données compréhensible par tous les outils et modèles mathématiques de la plateforme. Il permet de les paramétrer afin de les transmettre comme entrées aux modèles hydrologiques. Le module comporte les aspects suivants :

- L'extraction de sites hydrologiques :

Limitée pour l'instant aux cours d'eau et à leurs stations hydrométriques, l'extraction de sites hydrologiques doit permettre un choix interactif des sites dont les données hydrologiques sont éditables et les objets géographiques visualisables. Cette sélection peut être : simple quand elle se rapporte à un ou plusieurs cours d'eau (streams) ou biefs (reaches) ou aux tributaires (tributaries) d'un cours d'eau considéré ou encore à des stations hydrométriques (figure 4.6).
Ou plus complexe, par exemple pour réaliser l'extraction d'un cours d'eau (ensemble de biefs) entre deux stations hydrométriques (figure 4.7).

- L'extraction de données provenant d'un raster (grid) :

Elle permet d'extraire des séries d'informations spatialisées associées à des objets géographiques comme la pluie moyenne sur un bassin, le débit unitaire à l'endroit d'une station (figure 4.8), qui sont nécessaires au fonctionnement du modèle mis en œuvre.

- L'extraction de données spatialisées provenant de couches d'objets géographiques :

Elle offre la possibilité d'extraire des données contenues comme propriétés d'objets géographiques organisés sous forme de couche.

- L'extraction de données non spatialisées provenant de bases de données :

Cette fonctionnalité donne à l'utilisateur ou aux modules de traitements, la possibilité de se connecter à une base de données de son choix afin d'extraire des données selon des critères libres.

- Une interface de paramétrage des données des sites hydrologiques en entrée des modèles :

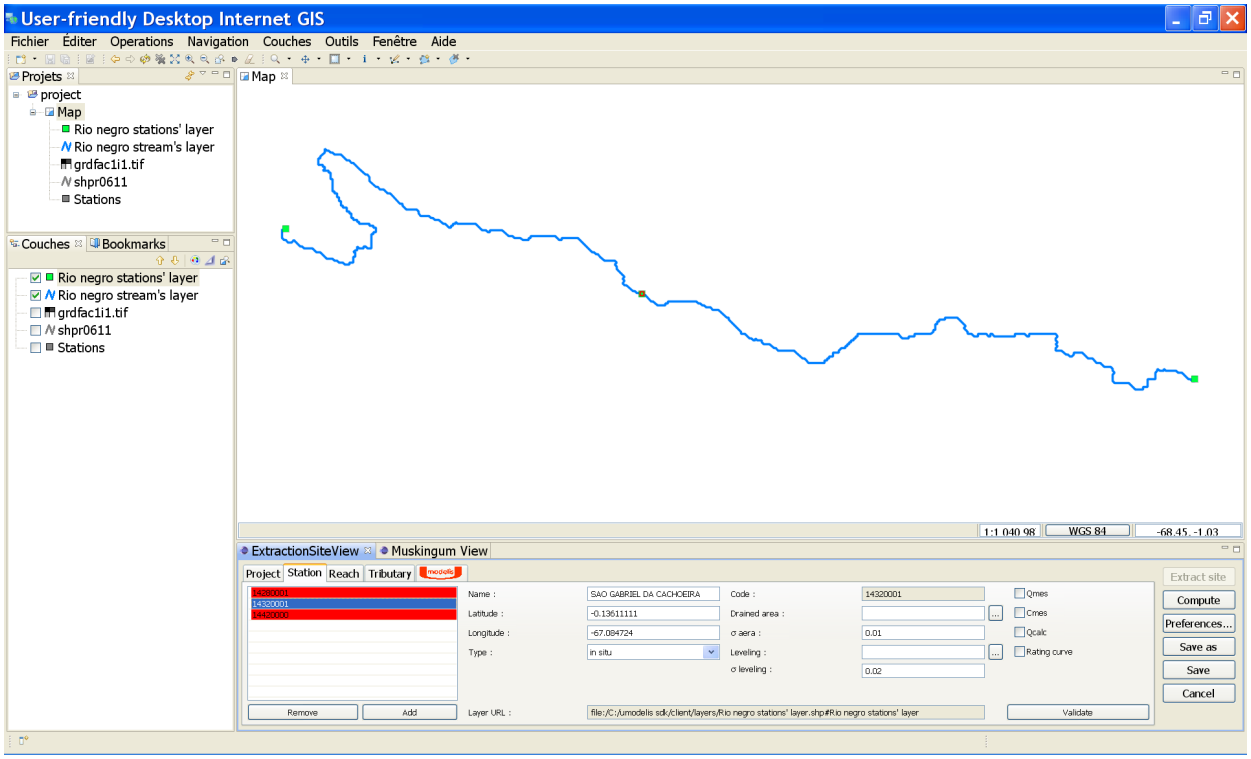

FIG. 4.6 : Capture d'écran d'un exemple de sélection d'une station avec l'interface d'édition de stations dans Umodelis.

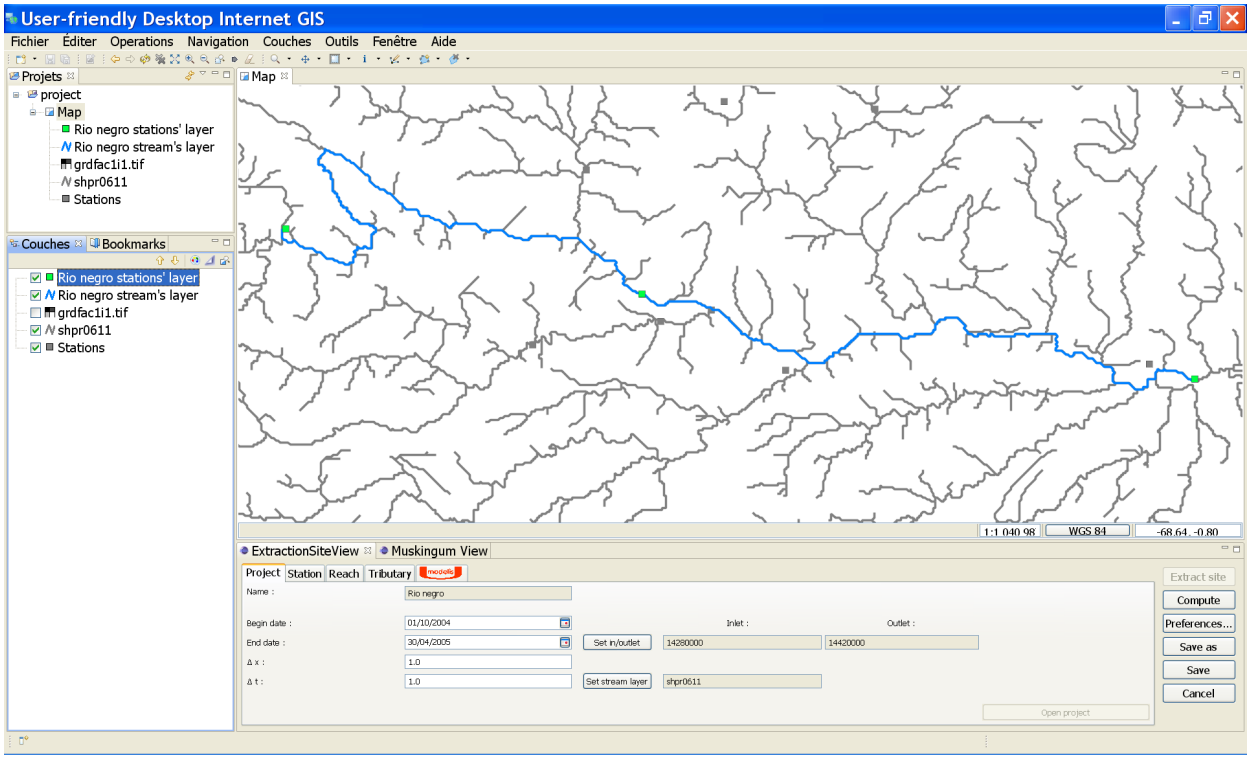

FIG. 4.7 : Capture d'écran du module de création de sites d'Umodelis après une extraction d'un cours d'eau (bleu) entre deux stations (vert).

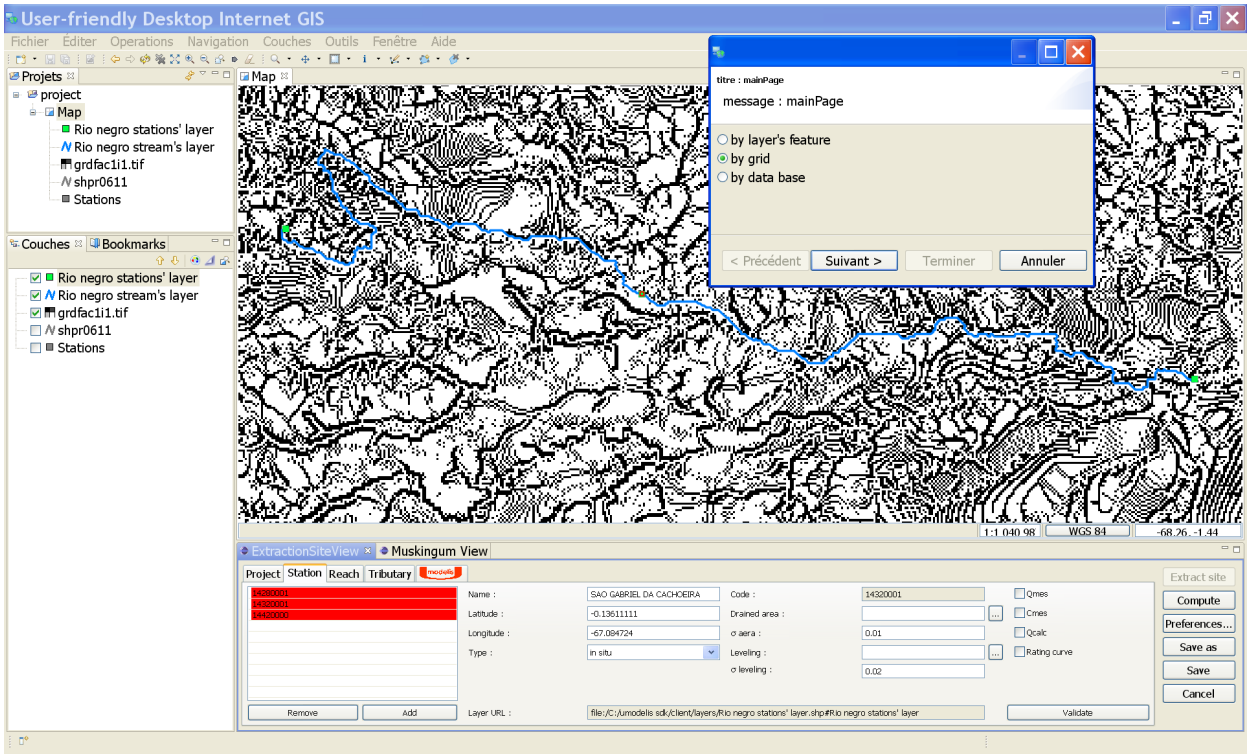

FIG. 4.8 : Capture d'écran d'un exemple d'extraction de données spatialisées raster (grid) sous Umodelis.

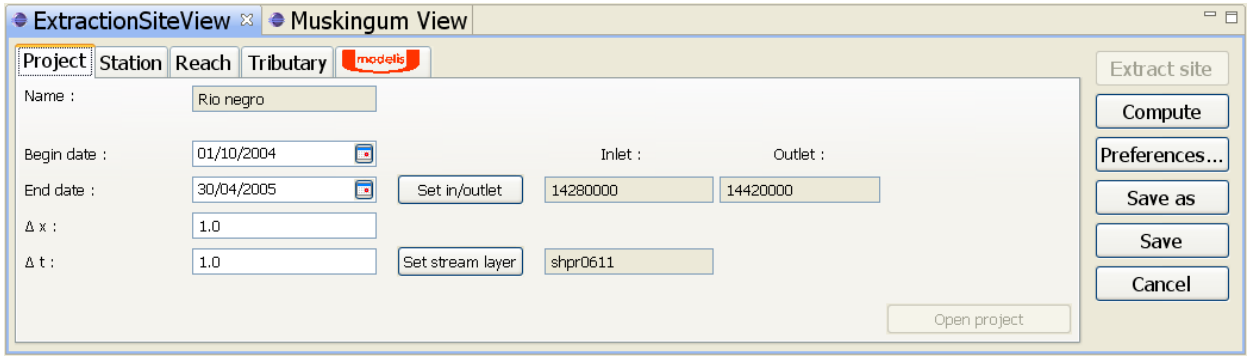

FIG. 4.9 : Capture d'écran de l'IHM d'édition de paramètres de modèles (projet) dans Umodelis.

Cette interface offre à l'utilisateur la possibilité de définir un ensemble de données et de paramètres en entrée des modèles. Ces données et paramètres appartiennent aux sites hydrologiques qui sont regroupés sous l'entité projet, tel qu'il a été défini à la section 3.4 page 15. Un projet est enregistrable, modifiable et distribué : accessible depuis n'importe quel poste, à travers le réseau (figure 4.9).

4.2.2.2 Exploitation de résultats

Deuxième outil intégré à la plateforme, le module d'exploitation des résultats de simulations (figure 4.10), offre l'interface de contrôle des simulations et présente, selon les critères de l'utilisateur, les données en sortie des modèles sous forme de graphiques et de tableaux. Les résultats sont enregistrables et distribués. Cet outil qui existe déjà sous forme d'application autonome, illustre la capacité de la plateforme à intégrer des outils existants.

4.2.3 Serveur de modélisation hydrologique

Conception et réalisation d'une structure d'accueil des modèles hydrologiques qui est centralisée, distribuée et qui offre la possibilité de contrôler les simulations à distance, capable de répartir les calculs et enfin d'intégrer les implémentations de modèles écrits dans différents langages (principalement Fortran et C). A titre d'exemple, le modèle Muskingum-Cunge est disponible pour le calcul de débit d'un cours d'eau par propagation.

4.3 Conclusion

Motivé par l'inadéquation des plateformes de modélisation existantes, le projet Umodelis constitue une base innovante en réponse aux besoins de la communauté scientifique. Le chapitre suivant va aborder la conception de l'architecture de ce projet.

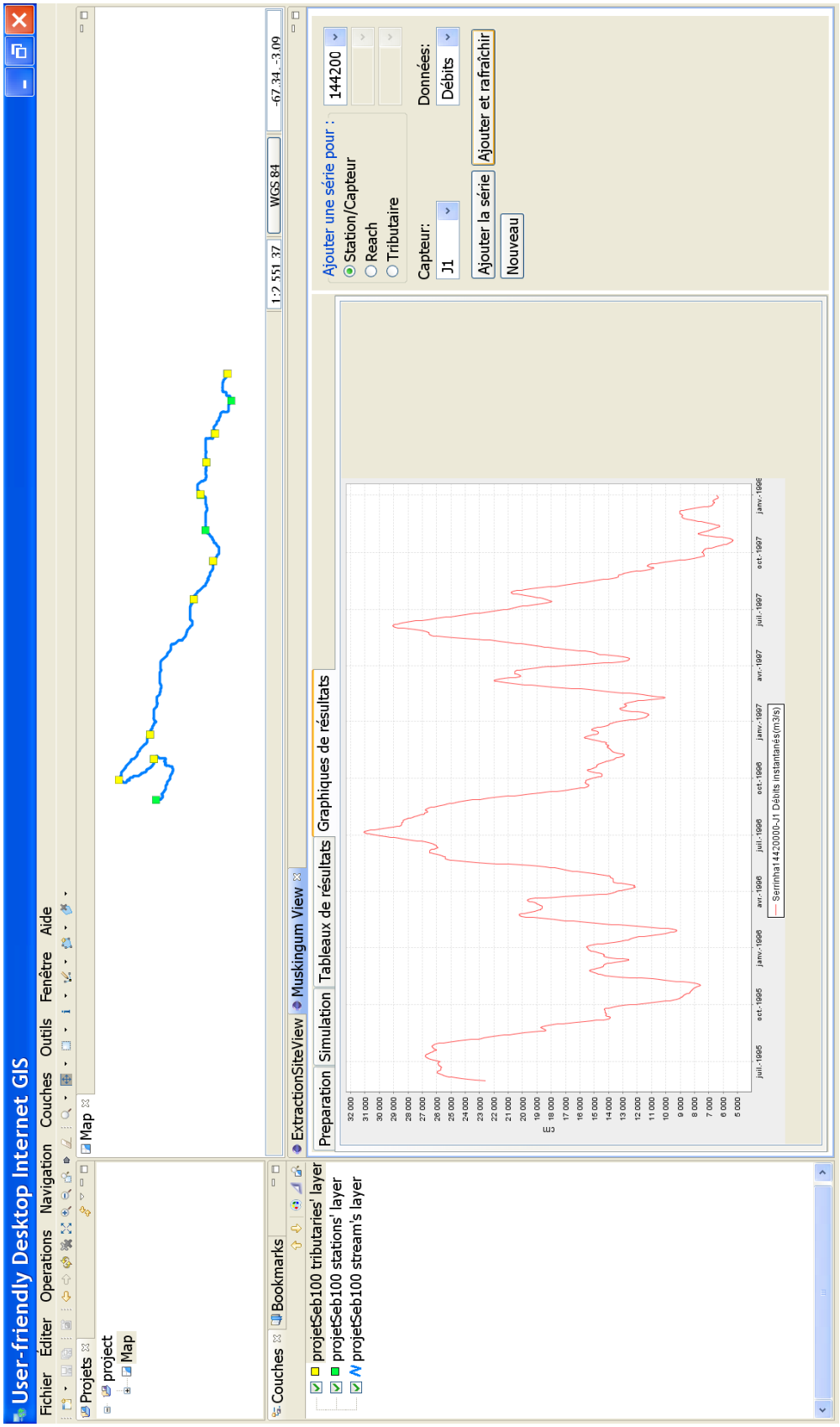

Fig. 4.10 : Capture d'écran du module de visualisation des résultats de simulations sous forme graphique dans Umodelis.

CHAPITRE 5

Architecture d'Umodelis

L'architecture de la plateforme Umodelis découle d'un travail mené en deux étapes : l'analyse et la modélisation. Après une courte réflexion sur le choix du langage de modélisation de l'architecture, le chapitre abordera l'analyse des exigences et des contraintes exprimées lors du chapitre 2 page 9. L'analyse se conclut par l'organisation en couches d'Umodelis et sur les orientations modulaires et distribuées de son architecture. Enfin, à partir de l'analyse, une modélisation de l'architecture est réalisée avec une présentation du point de vue fonctionnel et une autre orientée composants.

5.1 Choix du langage de modélisation logicielle : UML

Un de mes premiers objectifs lors de ce projet a été de mener une réflexion sur le langage de modélisation et de le promouvoir au sein de l'équipe. Unified Modeling Language (UML) est un langage incontournable du génie logiciel. L'Object Management Group (OMG) qui définit et maintient le standard UML, lui confère une certaine indépendance et pérennité. Il existe bien des alternatives comme le modèle entité-relation (Entity-Relationship Diagram ou ERD, affilié à Merise) mais j'ai considéré UML comme le plus adapté au projet Umodelis : ses diagrammes de cas d'utilisation et d'activité sont compréhensibles par les utilisateurs et les diagrammes de séquences, d'états, de composants et de classes décrivent tous les aspects du génie logiciel de façon relativement opérationnelle. D'autre part, l'adoption d'UML, assure une documentation de la plateforme compréhensible par de nombreux développeurs. Les diagrammes UML seront régulièrement utilisés pour illustrer la conception d'Umodelis.

5.2 Analyse des exigences et des contraintes

En traduisant les exigences et les contraintes exprimées par les utilisateurs du système, l'analyse cherche à dégager des spécifications afin de guider la modélisation de l'architecture. Ainsi, l'équipe a déduit l'orientation modulaire et distribuée du système. Par la suite, mon travail a consisté à proposer l'organisation en couches d'Umodelis qui est présentée en conclusion de l'analyse.

5.2.1 Architecture modulaire

Une des principales exigences est la capacité de la plateforme à intégrer les outils de traitements existants ainsi que les futurs travaux de recherche. Elle sous-entend un faible couplage entre la plateforme et les outils et entre les outils. Une modification de l'implémentation de la plateforme doit faiblement impacter les outils et inversement.
Isoler les outils revient à les considérer comme des composants du système utilisant des services proposés par la plateforme afin d'interagir avec leur environnement. Les communications entre la plateforme et les

outils et éventuellement inter outils sont nécessairement normalisées : les interfaces et les structures de données doivent être communes. Il en résulte une architecture basée sur la modularité des outils de traitements et la mise en place de services et de structures de données communes proposés par la plateforme.

5.2.2 Architecture distribuée

L'idée d'une architecture distribuée ou multi tiers provient de l'exigence de la distribution des données de simulation (entrées et résultats), de la factorisation des ressources de calcul sur un serveur ou une ferme de calcul et l'éventuelle répartition des calculs à l'intérieur de cette ferme. Elle s'appuie aussi sur la localisation des données : locales (systèmes de fichiers) ou distantes (serveurs cartographiques et bases de données). L'architecture doit résolument être distribuée.
Dans un premier temps, l'approche client-serveur a été considérée. Les itérations de l'analyse ont abouti au choix d'une architecture multi tiers, capable de fédérer les multiples sources de données et de les abstraire au client, ce dernier n'ayant qu'un seul interlocuteur : le serveur de modélisation hydrologique. Le serveur de modélisation réalise la centralisation des algorithmes des modèles hydrologiques et permet la distribution des calculs sur une ressource qui est mise à la disposition de tous les clients. Enfin, cette architecture permet la variété des formes de clients : aussi bien un client lourd intégré à un SIG qu'un client Web grâce à l'intégration d'un serveur Web au sein de l'architecture.
D'autre part, le serveur de modélisation doit assurer un certain niveau de sécurité, notamment gérer son accès et celui aux modules qu'il héberge et garantir un certain niveau de sûreté de fonctionnement concernant par exemple l'exécution des simulations : l'échec d'une simulation ne doit pas compromettre le fonctionnement du serveur. Les simulations devront être exécutées de façon indépendante et ne partager aucune donnée entre elles. Enfin, le serveur doit rendre les mécanismes d'accès et de persistance des structures de données transparents des moyens de stockage mis en œuvre.

5.2.3 Organisation en couches

La figure 5.1 fait le lien entre l'organisation logicielle classique en couches (ROQUES et VALLEE, 2004) et les constituants d'Umodelis. Une des particularités d'Umodelis réside dans l'utilisation d'un SIG pour l'implémentation de l'une de ses couches présentation (l'autre étant un client Web). Le SIG sert, en partie, d'IHM pour les modules de traitements afin d'accéder aux données spatialisées et de représenter les objets géographiques. En effet, la plateforme en s'appuyant sur le SIG délègue à celui-ci le rendu graphique ainsi que les interactions entre l'homme et la machine (sauf celles spécifiques aux modules de traitements). Elle utilise également les fonctionnalités avancées du SIG pour l'accès aux données et objets présents sur les systèmes de fichiers ou distants sur un serveur, généralement des serveurs de cartes comme ArcGIS server ou GeoServer qui sont accessibles par divers protocoles (voir section 6.2.2 page 35).
L'idée, comme elle a été formulée lors de l'état de l'art, est de concentrer le développement ultérieur de la plateforme sur les aspects simulation et traitements hydrologiques de la plateforme et de l'interfacer avec un SIG développé et maintenu par un tiers.
L'autre couche de présentation représentée par le client Web fournit un accès aux fonctionnalités des modules de traitements et une interface de contrôle des simulations mais n'offre pas la représentation des objets géographiques.
Concernant les autres composantas d'Umodelis, la couche contrôle est assurée par les modules de traitements, la logique métier est implémentée dans les modules et les modèles hydrologiques. Le serveur de

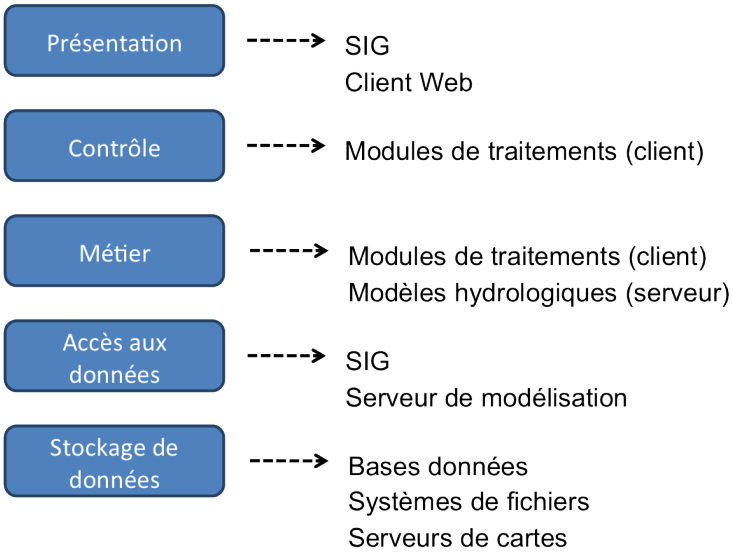

FIG. 5.1 : Organisation en couches de l'architecture d'Umodelis.

modélisation est chargé de l'accès aux données hydro-climatiques (spatialisées ou non), stockées dans les bases de données de l'IRD.

5.3 Modélisation de l'architecture

Sur la base des aspects modulaire et distribué de la plateforme, j'ai effectué la modélisation fonctionnelle de la plateforme puis la modélisation orientée composants.

5.3.1 Fonctionnelle

La figure 5.2 présente une vue fonctionnelle du système Umodelis. Les fonctionnalités du client SIG sont celles d'un SIG classique et feront l'objet d'une discussion au chapitre 6 page 34. La partie création de sites et les interactions graphiques avec les objets géométriques ne sont pas disponibles pour le client Web car il ne possède pas de couche SIG. Ainsi, le client Web ne représente pas graphiquement les sites hydrologiques mais affiche leurs caractéristiques alphanumériques : les objets hydrologiques. Cependant, le développement de bibliothèques graphiques Web orienté SIG ces dernières années (Google maps, etc.), donne la possibilité d'implémenter ultérieurement cette couche SIG dans le client Web. Cette étude fonctionnelle se concentre sur la partie réellement développée (nommée développement sur la figure 5.2) au cours du projet Umodelis. Elle est organisée en quatre sous ensembles : les services, les modules de traitements, les modèles hydrologiques et les flux de données.

5.3.1.1 Services

Les services comprennent les bibliothèques dont le premier élément, la bibliothèque SIG, est chargé de la représentation graphique des sites hydrologiques et des interactions avec le SIG (sélection, extraction d'objets géographiques et leurs propriétés, extraction de données rasters, persistance des objets géographiques,

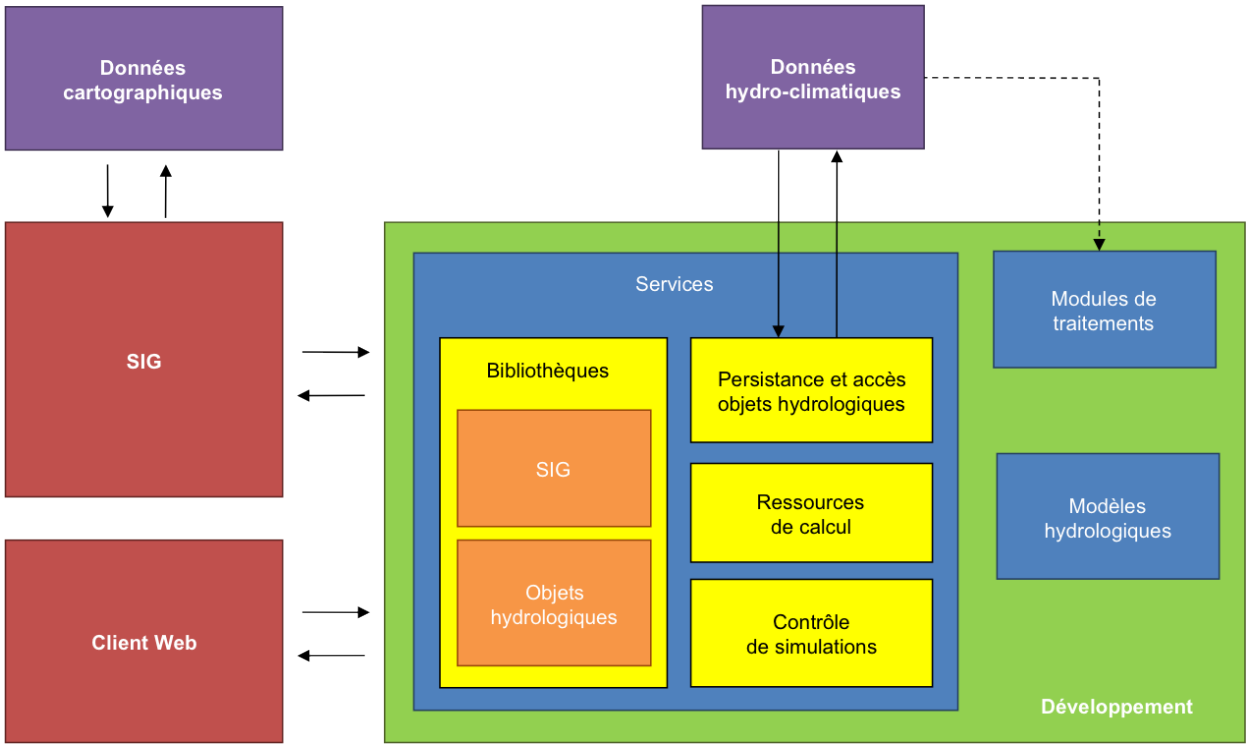

FIG. 5.2 : Schéma de l'architecture fonctionnelle d'Umodelis.

etc.). Une courte étude a permis de prototyper les fonctions et les structures de données classiques, communes à tous les SIG (création de couches, structures de données représentant les objets géographiques, etc.) afin de concevoir un ensemble d'interfaces utilisables par les modules, quel que soit le SIG choisi. Seule l'implémentation de ces interfaces est liée au SIG.

Le deuxième élément de la bibliothèque des services concerne les différents objets hydrologiques qui encapsulent les données hydrologiques spatialisées ou non. Ils sont modélisés et organisés sous forme de classes d'objets (au sens OMT) afin qu'elles soient utilisables par n'importe quel outil intégré à la plateforme. Ces objets tels que les stations hydrométriques ou les biefs, assurent la cohérence des concepts manipulés par les différents modules de la plateforme et rendent possible la communication entre les modules. L'ajout de nouveaux concepts géomatiques nécessaires à l'utilisation de nouveaux modèles hydrologiques est directement traduit par la simple création de nouvelles classes d'objets. L'étude et le lien entre ces deux bibliothèques seront présentés au cours du chapitre 8 page 47.

La puissance de calcul offerte par le service de ressources de calcul est un service matériel utilisable par les algorithmes de modélisation hydrologiques. En effet, les modèles hydrologiques sont centralisés sur une entité de calcul (simple serveur ou ferme de calcul). Ainsi, l'utilisateur pilote à distance les simulations. Le calcul est potentiellement réparti (load balancing) si cette entité de calcul est un ensemble d'unités de calcul connectées (ferme de calcul). L'autre avantage de la centralisation des modèles est de faciliter leur mise à jour par les équipes de recherche.

Le service d'accès et de persistance des objets hydrologiques assure l'accessibilité des données hydrologiques en entrée et résultats de simulations à travers le réseau.

Umodelis fournit un mécanisme de contrôle des simulations et de visualisation de leur état. Associé à une

IHM, ce mécanisme permet à l'utilisateur de connaître l'opération qui est en cours, l'état d'avancement de la simulation et de l'interrompre éventuellement.

5.3.1.2 Modules de traitements

Sous cette entité sont regroupés les modules de traitements de données. Ces modules ont pour rôle de préparer les données en entrée des modèles, de manipuler des concepts géographiques ou d'exploiter les résultats de simulations. Concernant le projet Umodelis, il a été décidé d'implémenter le module de création de sites hydrologiques et d'intégrer un outil existant d'exploitation de résultats de simulations. Leur description a été donnée à la section 4.2.2 page 22. Les modules sont des utilisateurs des services de fonctions SIG, d'accès et persistance des données, de contrôle de simulations et manipulent les objets hydrologiques.

5.3.1.3 Modèles hydrologiques

De la même façon, les modèles hydrologiques sont regroupés sous cette appellation. Ils sont la logique de la simulation, contiennent la valeur scientifique et sont responsables de la transformation des données d'entrées en résultats de simulation. Les modèles utilisent les ressources de calcul et les objets hydrologiques.

5.3.1.4 Flux de données

Concernant les flux de données (flèches sur la figure 5.2), l'accès aux données hydro-climatiques est géré par le service d'accès et de persistance. Toutefois, un lien en lecture seule reste encore possible entre les modules de traitements et les données (lien en pointillé). Par contre, l'accès aux données cartographiques (objets géographiques, leurs propriétés et autres données spatialisées) s'effectue au travers des services proposés par le SIG afin de profiter de toutes ses fonctionnalités. Seule la persistance des objets géographiques est du ressort du SIG, les objets hydrologiques (données spatialisées ou non) créés ou modifiés par les modèles sont sauvegardés par le serveur de modélisation.

5.3.2 Orientée composant

La figure 5.3 représente l'organisation d'Umodelis sous forme de composants UML. L'intérêt de cette représentation est d'illustrer l'architecture n tiers et l'orientation composants d'Umodelis : le fragment UML client (client web et SIG), le composant serveur de modélisation et le composant bases de données représentant toutes les bases de données du système. Ce diagramme s'attache aussi à faire ressortir les interfaces inter composants (les interfaces intra composant seront décrites dans les chapitres dédiés aux modules).

5.3.2.1 Clients

Le fragment UML modules regroupe les différents outils de traitements conçus sous forme de composants du système. L'intégration de nouveaux outils se fait donc par l'ajout de nouveaux composants respectant un patron de conception. L'aspect modulaire énoncé dans l'analyse de l'architecture est ainsi implémenté. Ces composants ont à leur disposition des interfaces afin d'interagir avec le SIG pour les fonctionnalités concernant les objets géographiques et avec le serveur de modélisation pour l'accès aux objets hydrologiques, leur persistance ainsi que le contrôle des simulations. Un soin particulier est apporté à l'interface du SIG qui, comme précédemment dit, doit assurer l'indépendance des modules vis à vis du SIG choisi. Enfin, les

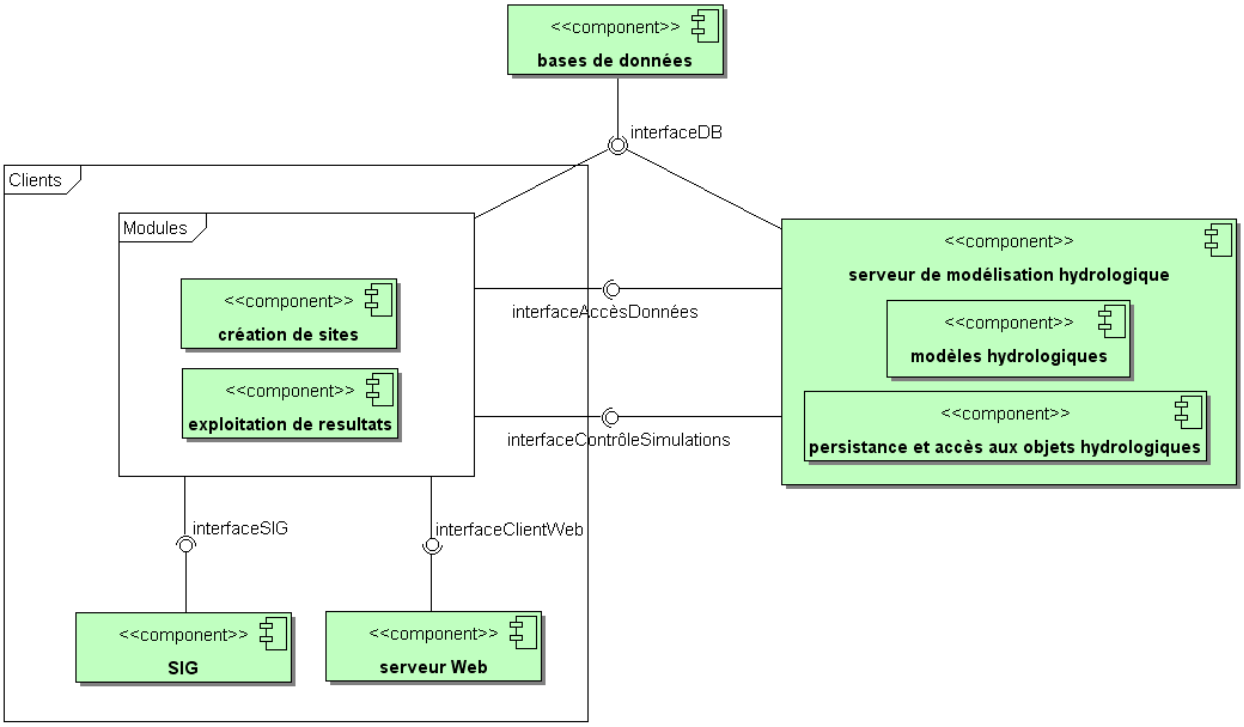

FIG. 5.3 : Diagramme de composants UML du système Umodelis et de son environnement.

modules présentent une interface permettant l'invocation de leurs fonctions à travers le Web : les requêtes émises par le client Web sont interprétées par un serveur Web puis relayées aux modules.

5.3.2.2 Serveur de modélisation hydrologique

Le serveur de modélisation est un composant lui même formé à partir de composants tels le composant de persistance et d'accès aux objets hydrologiques et celui qui encapsule les modèles hydrologiques. Là encore, l'ajout d'un nouveau modèle est simplifié par l'architecture modulaire du serveur et le respect d'un patron de conception. Le composant de persistance et d'accès, chargé d'interroger et de modifier les bases de données distribuées, a aussi pour mission d'encapsuler les données extraites des bases de données en instance de classes d'objets hydrologiques définis dans la bibliothèque des services. Ce module assure donc l'abstraction du stockage des données et de l'uniformisation de son format au sein de la plateforme.

5.3.2.3 Les bases de données

Le composant bases de données représente toutes les bases de données de l'IRD concernant les données hydro-climatiques du bassin amazonien. Elles sont implémentées sous différents SGBDR (PostGreSQL et MySQL) et présentent les pilotes permettant l'accès à leurs données. Cet accès est géré par le serveur de modélisation afin d'abstraire aux modules les différents paramètres de connexion et techniques de transaction mais l'accès direct des modules de traitements aux bases de données n'est pas interdit et est par ailleurs prévu dans la conception du module de création de sites hydrologiques.

5.4 Conclusion

L'architecture d'Umodelis présente un caractère fortement modulaire s'appuyant sur des interfaces et des services au sein d'une architecture distribuée. Sa principale propriété est le découplage des couches logicielles : la couche présentation, afin de garantir la plus forte indépendance avec le SIG utilisé, la couche métier avec la logique des modules implémentée dans le client et la logique des modèles hydrologiques centralisée sur le serveur de modélisation, la couche accès aux données avec l'accès et la persistance des objets hydrologiques par le serveur de modélisation et l'accès et la persistance des objets géographiques par le SIG et enfin la couche stockage de données qui est implémentée par les bases de données de l'IRD et les serveurs de cartes. Avant la présentation des points importants de l'architecture de la plateforme, la suite du mémoire a pour sujet le choix du SIG ainsi que le langage d'implémentation d'Umodelis.

CHAPITRE 6

User-friendly Desktop Internet GIS

L'idée d'intégrer un SIG dans le système Umodelis est d'éviter de réinventer ce qui existe déjà et de bénéficier de l'expertise et la maintenance d'un tiers. Au cours de ce chapitre, un bilan sur les besoins fonctionnels en SIG est donné afin d'éclairer le choix du logiciel uDig présenté en seconde partie. Une discussion sur les alternatives libres et commerciales à uDig complète ce chapitre.

6.1 Bilan des besoins en SIG

L'étude préliminaire a permis d'établir les besoins fonctionnels fondamentaux SIG pour le développement du projet Umodelis.

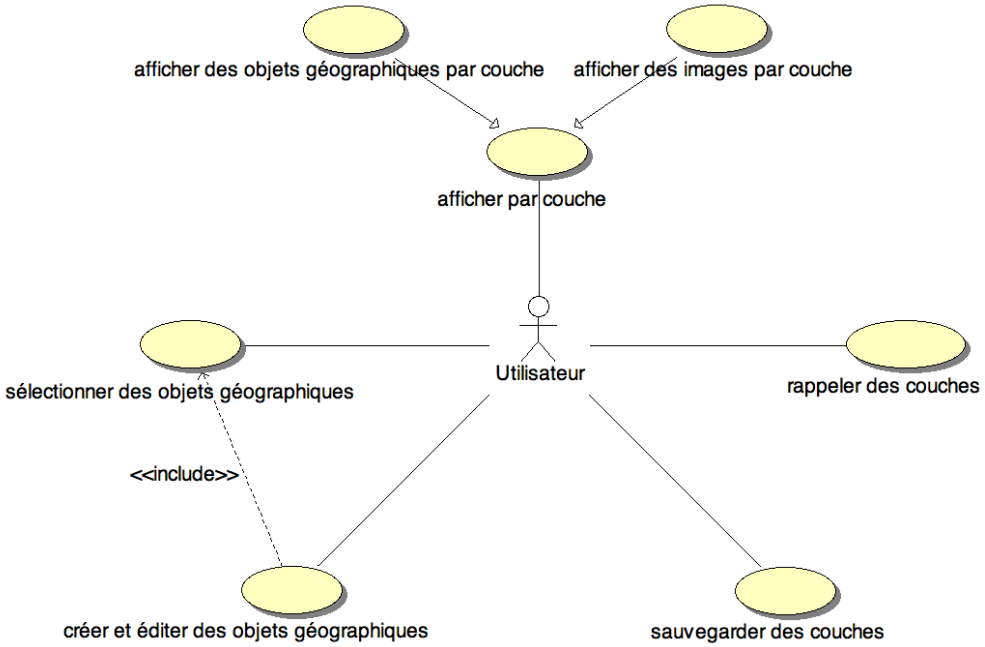

FIG. 6.1 : Diagramme de cas d'utilisation UML d'un client SIG.

Le SIG choisi doit nécessairement présenter les fonctionnalités de base des SIG illustrées sur la figure 6.1, à savoir l'affichage sous forme de couches d'images aux formats les plus répandus (TIFF, JPEG, formats SIG propriétaires, etc.) et l'affichage de couches d'objets géographiques de même nature et leurs propriétés en proposant plusieurs systèmes de projection de coordonnées, la création et l'édition d'objets géographiques ainsi que la sauvegarde et le rappel des couches. Cet ensemble de cas s'accompagne des habituelles IHM et fonctions d'exploitation graphique comme la sélection des objets à l'écran, le zoom, etc.

Selon les contraintes exprimées au chapitre 2 page 9, la licence du logiciel doit être gratuite (GPL ou LGPL) avec de raisonnables garanties de pérennité, portable multi systèmes d'exploitation (au moins Windows, Mac OS X et Linux) et son implémentation doit respecter les standards de l'OGC.
Deux points sont essentiels quant à l'intégration du SIG dans la plateforme (voir chapitre 5 page 27). Le premier concerne l'extension du SIG : il doit être facilement évolutif par ajout de modules, qui correspondent à ceux prévus par l'architecture d'Umodelis. Le deuxième est l'extraction de données cartographiques provenant de différentes sources comme les serveurs de cartes ou les bases de données spatialisées qui doivent également respecter les standards existants.

6.2 uDig

Parmi les quelques candidats répondant aux besoins exprimés ci-dessus uDig présente un intérêt particulier.

6.2.1 Histoire

Né à l'initiative du Canadian GeoConnections program et financé au début par celui-ci, le projet uDig voit le jour au printemps 2004. Depuis 2005, il se poursuit comme projet indépendant soutenu par Refractions Research (communauté de développement). Ce projet bénéficie d'utilisateurs prestigieux comme le cite RAMSEY (2006) : United Nation Food & Agriculture Organization (UNFAO), US State Department (figure 6.2), British Columbia Ministry of Forests et d'une communauté de développeurs qui continue de grandir. uDig est distribué sous licence LGPL.

6.2.2 Description fonctionnelle

Bien entendu, uDig possède les fonctionnalités énumérées dans le bilan précédant. La gestion et la création de couches d'objets géographiques qui sont appelés features, sont étoffées. L'affichage est réactif et propose de nombreux systèmes de projection. La possibilité de lire les formats d'images les plus répandus et d'importer des couches de données cartographiques venant de logiciels commerciaux ou libres donne une dimension universelle. Mais, la richesse d'uDig, réside dans l'accès facile et transparent à la donnée cartographique :
La superposition des schémas des figures 6.3 et 6.4 montre respectivement la correspondance entre les standards de l'OGC et les applications les supportant proposées par la fondation Geospatial Open Source (OSGeo). uDig (symbole moniteur et la Terre) accède aux données cartographiques soit en passant par un serveur de cartes par exemple Geoserver selon le protocole Web Feature Service (WFS) soit directement par requêtes Simple Feature – Structured Query Language (SF-SQL) aux bases de données compatibles dont PostGIS. Les schémas ne montrent pas l'accès aux données sur les systèmes de fichiers mais il est effectivement prévu dans uDig par l'intermédiaire du menu « fichier » classique.
Par contre, l'offre d'outils de traitements n'est pas étoffée. En effet, uDig se positionne avant tout comme une base offerte à la communauté géoinformatique pour le développement d'applications SIG avancées (GARNETT, 2005). Les développeurs peuvent compter sur l'aspect modulaire d'uDig qui repose sur la plateforme Eclipse Rich Client Platform (Eclipse RCP) et la bibliothèque GeoTools.

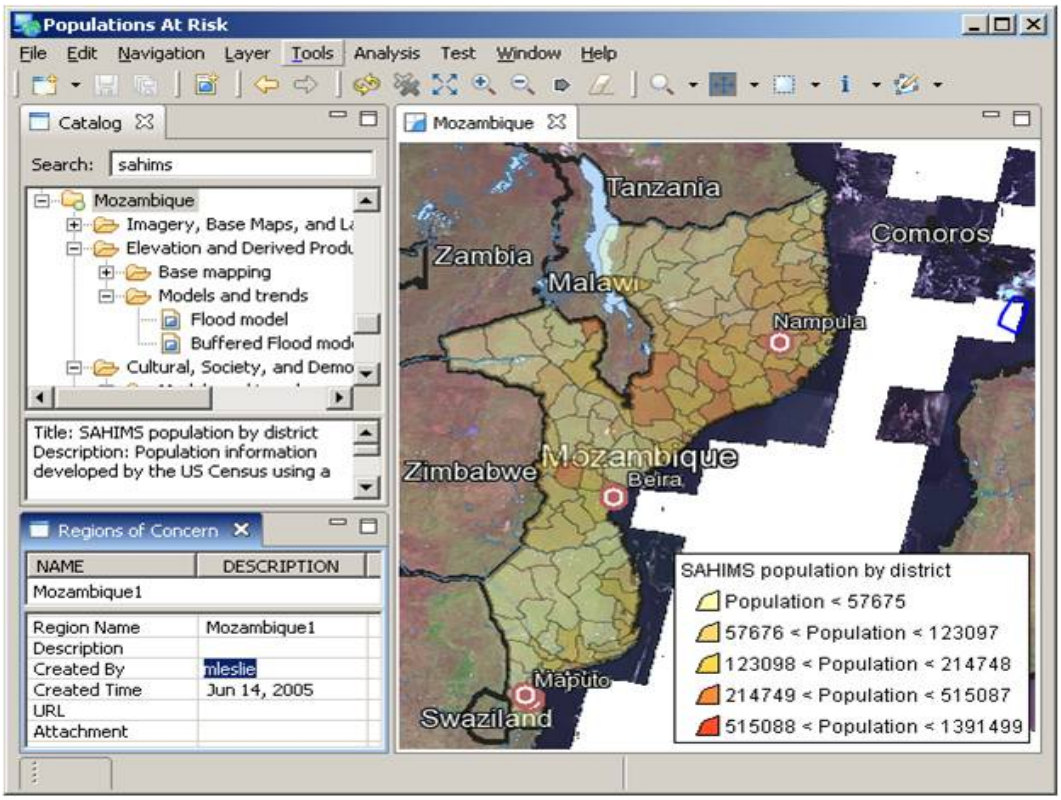

FIG. 6.2 : Capture d'écran d'uDig, population at risk US State Department (d'après RAMSEY, 2006).

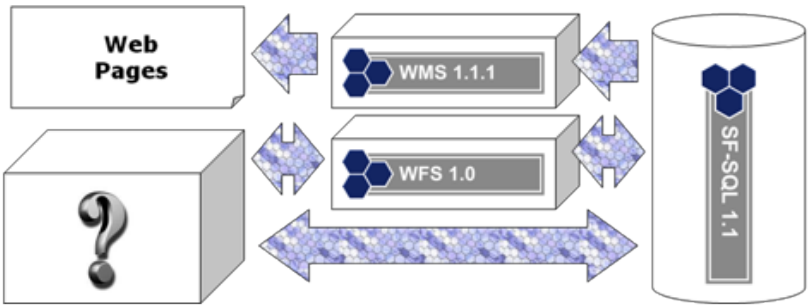

FIG. 6.3 : Architecture des standards OGC (d'après RAMSEY, 2006).

FIG. 6.4 : Schéma d'accès aux données spatialisées dont cartes via serveur (d'après RAMSEY, 2006).

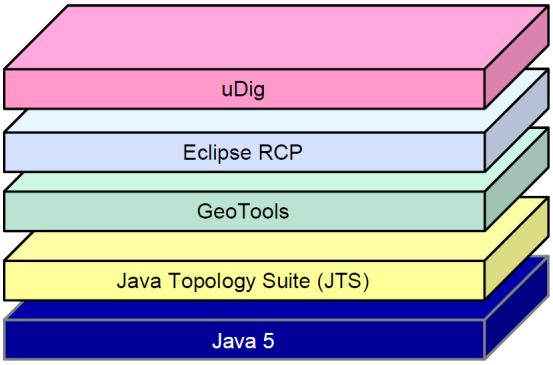

FIG. 6.5 : Architecture uDig en couches (d'après GARNETT, 2005).

6.2.3 Architecture

La figure 6.5 situe la place d'Eclipse RCP et de GeoTools dans la pile de bibliothèques d'uDig.

6.2.3.1 Eclipse RCP

Eclipse RCP constitue la pierre angulaire d'uDig. En effet, suivant la figure ci-dessous (6.5), uDig est une extension de la plateforme générique (UI ou User Interface sur la figure) d'Eclipse. Rappelons qu'Eclipse RCP est à l'origine un projet d'IBM qui l'a depuis donné à la communauté du libre en l'organisant autour d'une fondation. IBM souhaitait factoriser ces environnements graphiques (Graphical User Interface ou GUI) sous la forme d'une base commune sur laquelle développer ses applications commerciales. Son langage d'implémentation, Java (Sun Microsystems), assure une portabilité multi OS (Solaris, Windows, Mac OS X, Linux, etc.). Eclipse est distribué sous licence gratuite Eclipse Public License (EPL).
Cette plateforme générique repose sur les technologies telles que JFACE et Standard Widget Toolkit (SWT) pour les composants graphiques (widgets) ainsi que le système de gestion dynamique de modules (cycle de vie des modules) de l'Open Services Gateway initiative (OSGi).
IBM a souhaité que l'architecture soit la plus modulaire possible afin d'ajouter la logique métier de ses logiciels sous forme de modules. Ainsi Eclipse RCP propose diverses formes de points d'extension (plugins extension points), allant du simple bouton à l'éditeur. Les gestionnaires d'événements sont aussi considérés comme des points d'extension.
La place des modules de traitements d'Umodelis se trouve au niveau de la couche « Extensions » sur la figure 6.6. On retrouve ainsi une architecture imbriquée : Les modules d'Umodelis sont des extensions d'uDig qui est lui-même une extension d'Eclipse RCP.

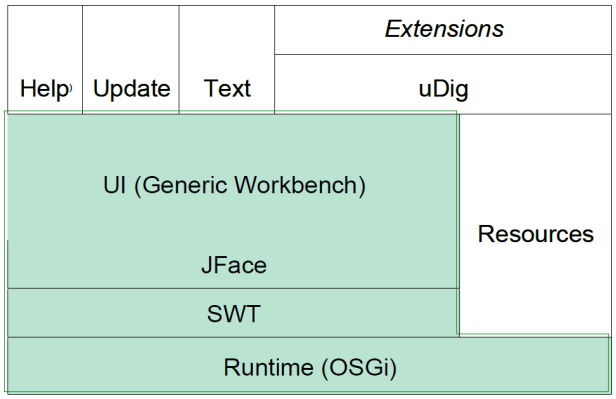

FIG. 6.6 : Architecture en couche d'Eclipse RCP (d'après GARNETT, 2005).

6.2.3.2 Getools

Geotools est une bibliothèque Java de l'OSGeo s'appuyant sur Java Topology Suite (JTS) de Sun Microsystems pour la représentation graphique. C'est une spécification d'un standard géomatique dont uDig apporte une implémentation encore incomplète mais nettement suffisante pour Umodelis. Parmi les très nombreuses spécifications, on note :

- Une structure pour les formes géométriques :

Les formes géométriques utilisées dans les SIG comme le point, la ligne par interpolation linéaire de points (line string), l'anneau (linear ring) ou le polygone (polygon) sont spécifiées.

- Une structure de système de projection de coordonnées géographiques.
- Une structure d'objets géographiques :

Les objets géographiques sont encapsulés dans une structure appelée feature qui contient essentiellement le système de projection de coordonnées et la géométrie de l'objet qui est composée de coordonnées.

- Une structure contenant les features appelée couche (layer) elle-même contenue dans une carte (map).
- Un ensemble de spécifications pour le rendu graphique des features.
- Des fabriques d'instances (design pattern factory ; GAMMA *et al.*, 1994 ; voir annexe A.5 page 107) associées aux diverses structures.

Cette bibliothèque comporte aussi l'outillage pour les tâches les plus génériques parmi lequel un système de filtres de features ou une bibliothèque de fonctions d'analyse de rasters.
La version d'uDig (1.1 Release Candidat 12) utilisée lors du développement d'Umodelis n'était pas exempte de bugs toutefois le choix de cette pré version 1.1 nous préparait aux avancées de la version stable 1.1 censée corriger les erreurs de conception de la première version d'uDig.

6.2.3.3 Java

Enfin, le socle de base de la pile d'uDig est la machine virtuelle Java (Java Virtual Machine ou JVM) de Sun Microsystems qui interprète le pseudo code compilé Java en instructions natives. Le langage Java utilisé par l'ensemble de la pile et des points d'extension d'uDig explique le langage d'implémentation d'Umodelis. Java, étant orienté objet, il convient pour implémenter les modèles UML issus du travail de conception de la plateforme. D'autre part, l'équipe de développement possèdait, avant le démarrage du projet, une solide expertise en Java.

uDig s'inscrit dans une capitalisation de projets qui se concentrent sur leurs propres objectifs. Sa pérennité en est d'autant accrue : la division de l'information géographique en une chaîne permet la répartition de l'effort de travail sur plusieurs communautés de développement. De plus, uDig a le soutient d'une institution du Canada et voit son travail repris par un nombre grandissant de projets. La partie suivante place uDig face à des alternatives venant du libre et des logiciels commerciaux.

6.3 Alternatives

Trois autres SIG libres présentent une alternative à uDig : Quantum GIS (QGIS), OpenJump et generalidad valenciana SIG (gvSIG). Cette section montre les arguments qui ont prévalus au choix d'uDig mais il est important de savoir qu'il a été fait mi 2007. Par conséquent, certains arguments ne sont plus valables aujourd'hui. Les conséquences du choix du SIG sont cependant atténuées par le découplage mis en place entre Umodelis et le SIG utilisé (thème développé dans les chapitres 7 et 8, respectivement pages 42 et 47).

SOLUTION	QUANTUM GIS	GVSig	Openjump	UDig
Installation Pré-requis	Système: Windows, Mac, Linux	Système: Windows, Mac, Linux	Système: Windows, Mac, Linux	W-M-L
Commentaire	Installation simple par fichier exécutable	Installation simple par fichier exécutable	Installation simple par fichier exécutable	Simple exe
Formats Vecteur supportés	shape, mif/mid, tab, gml, txt			gml, Shapefile
Formats Raster supportés	Liste importante			TIF, JPEG, PNG, GIF
Connections SGBD supportées	PostGIS	PostGIS, Oracle		PostGIS, DB2, Oracle, MySQL
Connections aux services de cartographie en ligne supportées	WMS, WFS par un plugin (lecture seule)	WMS, WFS, WCS		WMS, WFS (lecture, écriture)
Outils d'édition de données vectorielles	Oui (+ outil de snapping et topologie **très** simple)		Oui	Oui (+ outil de snapping)
Export au format image	Oui	Oui	Oui	Oui, PNG, TIFF, GIF
Export au format PDF	Oui	Oui	Oui	Oui
Langue du logiciel	Français	Français	français	Français
Documentation (langue et qualité)	Anglais, Français, Workshop en français [http://softlibre.gloobe.org/doku.php/qgis/start]	Espagnol, Anglais	Anglais	Anglais
Activité de la communauté	Développement de plugins en python, documentation, workshop			
Language de programmation	C++ / Plug-in Python	Jython, Java, Groovy	Java	Java/Eclipse
Remarques	Possibilités de mise en page limitées. Peut être utilisé comme bibliothèque Python pour développer des applications SIG	Bonnes possibilités de mise en page	Très complet et robuste en édition de shapefile.	
Version mobile	Oui	Oui	Non	Non

FIG. 6.7 : Tableau de comparaison des SIG libres (d'après le site Web OSGeo).

6.3.1 Alternatives libres

La figure 6.7 offre un récapitulatif des fonctionnalités des SIG libres.

6.3.1.1 gvSIG

gvSIG est un projet local à la communauté autonome de Valence, donc fortement orienté par ses problématiques. Il est donc risqué de s'intégrer dans ce SIG qui pourrait prendre une orientation propre peu compatible à celle Umodelis.

6.3.2 OpenJump

Java Unified Mapping Platform (JUMP) est un projet abandonné qui a retrouvé un second souffle après son ouverture sous la forme du projet open source : OpenJump. Au commencement du développement d'Umodelis (mi 2007), JUMP était en pleine transition et son avenir était encore incertain, ce qui explique son rejet par l'équipe. Aujourd'hui ses fonctionnalités intéressantes en font un parfait remplaçant. OpenJump qui est écrit en Java collabore activement au développement de GeoTools. Sa bibliothèque graphique se retrouve dans GeoTools qui est utilisé par uDig.

6.3.3 Quantum GIS

QGIS est une surcouche du plus ancien SIG libre : Geographic Resources Analysis Support System (GRASS). Très riche en fonctionnalités, QGIS se différencie principalement d'uDig par son support partiel aux standards de l'OGC et son langage d'implémentation C++. Les extensions peuvent être également codées en

Python. QGIS possède une meilleure réactivité graphique que les autres SIG écrits en Java car il est compilé nativement et n'est pas interprété comme Java. Si QGIS est aussi intéressant qu'uDig comme client SIG pour Umodelis, il n'a pourtant pas été choisi, principalement à cause du langage d'implémentation des extensions : l'équipe de développement ne possède pas une grande expertise du langage C++ réputé plus difficile et source d'erreurs que Java.
Enfin, le choix d'uDig s'explique par la capitalisation d'expériences au cours d'un projet antérieur (Contribution de l'Altimétrie Spatiale à l'Hydrologie ou CASH) et par l'extensibilité remarquable de sa base Eclipse RCP.

6.3.4 Alternatives commerciales :

Bien que l'emploi d'un SIG commercial soit contraire aux contraintes d'Umodelis, il est intéressant de pouvoir comparer uDig à la solution SIG la plus répandue : ArcGIS de la société Environmental Systems Research Institute (ESRI).
Tout d'abord, ESRI fournit un support technique aux clients alors que celui-ci est inexistant chez Refractions Research qui, comme la plupart des distributeurs de logiciels libres, n'offre pas directement de support. Toutefois, l'utilisation et le développement des logiciels libres s'articulent autour de communautés à l'efficacité parfois comparable ou d'entreprises tiers qui proposent des services payants.
Bien sûr, ArcGIS possède plus d'outils qu'uDig, une longue expérience dans la géomatique et de nombreuses extensions (rien de comparable à Umodelis). Cependant, ESRI n'a toujours pas prévu le support de modules écrits en Java.

6.4 Conclusion

De par ses fonctionnalités, même limitées, son accès transparent et distribué aux données, sa conception modulaire robuste s'appuyant sur Eclipse RCP et divers standards, uDig constitue une base parfaitement conçue et pérenne pour Umodelis. Si le projet Umodelis a retenu uDig face à des alternatives pourtant toutes aussi valables, il doit limiter sa dépendance au SIG afin de permettre d'en changer en minimisant l'impact sur le reste de la plateforme. Ce thème ainsi que l'intégration du client web seront l'objet du prochain chapitre.

CHAPITRE 7

Intégration SIG et client Web

Après avoir défini l'architecture d'Umodelis et les besoins en SIG puis choisi uDig, nous allons nous intéresser à l'intégration du SIG et du client Web dans l'architecture d'Umodelis, tâche qui m'a été confiée. L'analyse de cette problématique conduit à l'application du patron de conception Modèle Vue Contrôleur (MVC) à la plateforme. Une modélisation UML est proposée afin d'illustrer la solution retenue.

7.1 Analyse

La problématique est la suivante : comment coupler un SIG et un client Web, deux couches de présentation de nature différente et de fonctionnalités différentes, avec les modules de traitements de la plateforme Umodelis. De plus, la dépendance devra être la plus faible possible : le code dépendant du SIG et du client Web devra être encadré et limité en volume.

Une solution classique pour organiser la couche présentation consiste à appliquer le patron MVC. Ce patron a pour but de séparer l'interface graphique de la logique de contrôle des requêtes utilisateur sur cette interface et du traitement de ces requêtes. Un bref rappel est fait sur les principes du MVC ainsi que sa variante MVC2 avant de décrire son application à notre problématique.

7.1.1 Rappels MVC

On distingue deux versions du patron de conception MVC : la version Reenskaug, adaptée aux applications non distribuées et la version model 2 adaptée aux architectures n-tiers.

7.1.1.1 MVC Reenskaug

Version développée par REENSKAUG (1979, 2003), elle organise la séparation de l'interface graphique dénommée vue de la logique de contrôle appelée contrôleur et des traitements des requêtes utilisateur, le modèle. Suivant la figure 7.1, les requêtes de l'utilisateur effectuées sur la vue sont transmises au contrôleur adéquat parmi un ensemble de contrôleurs. Le contrôleur analyse la requête afin de l'orienter vers le traitement approprié qui est encapsulé dans le modèle. Le modèle prévient toutes les vues qui lui sont associées d'une modification de son état ou de l'état de ses données, à charge aux vues d'afficher les mises à jour. Les mécanismes d'interception de requêtes et de notification de changement d'état sont le plus souvent conçus par programmation événementielle en utilisant le patron de conception observateur (design pattern observer ; GAMMA *et al.*, 1994 ; voir annexe A.8 page 109).

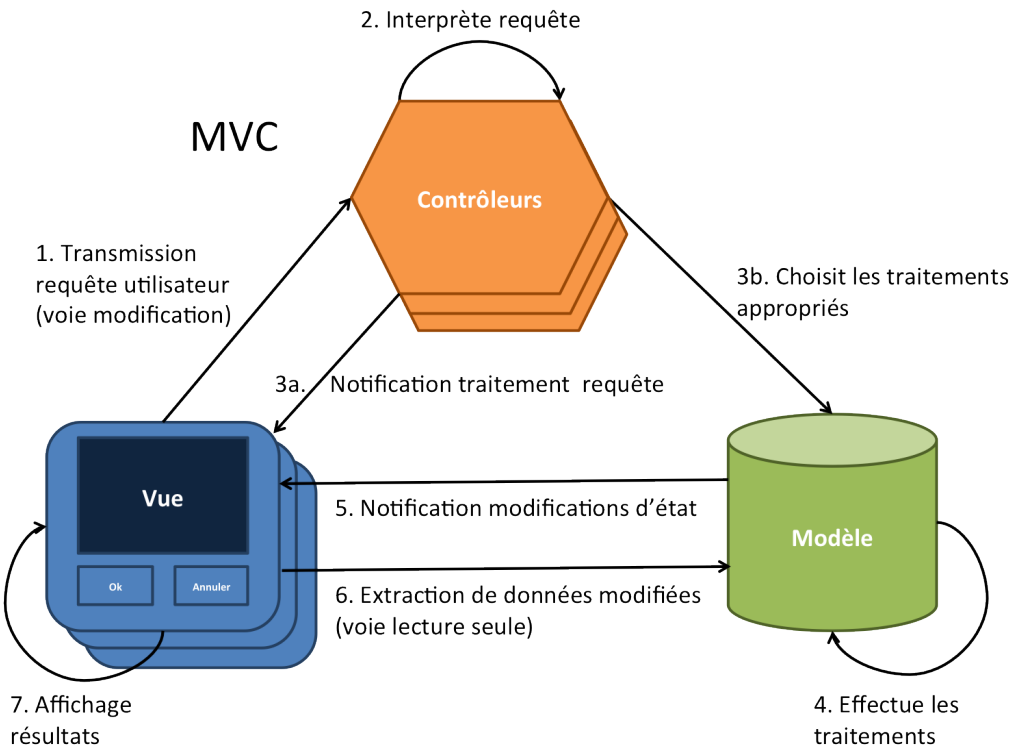

FIG. 7.1 : Schéma du patron de conception MVC Reenskaug.

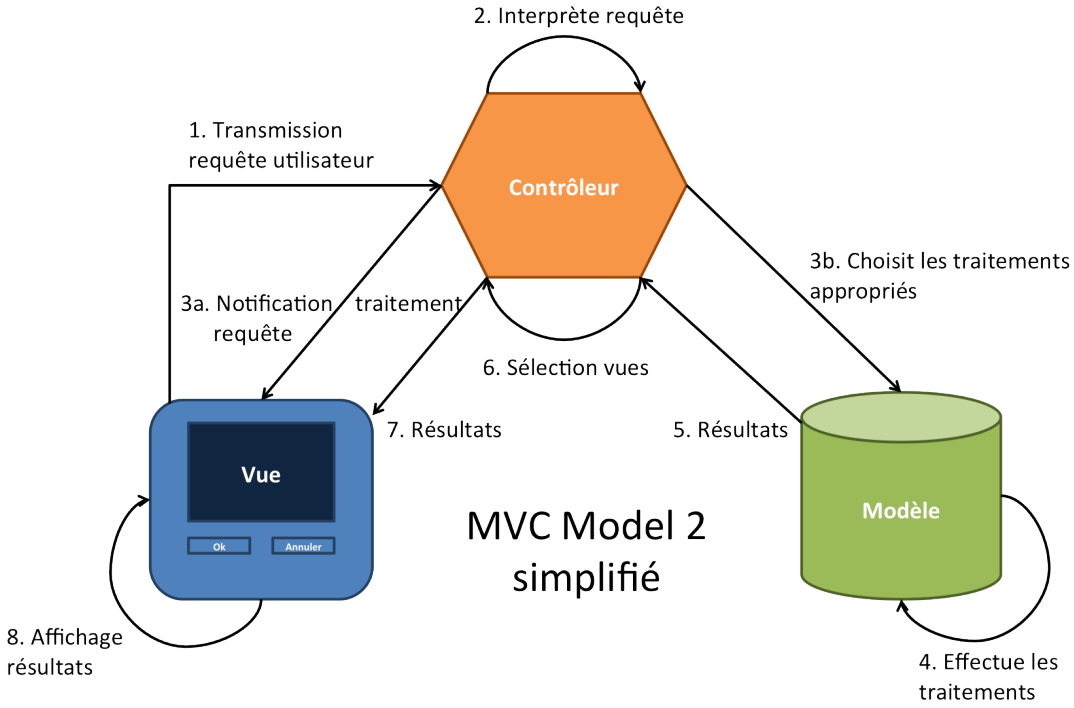

FIG. 7.2 : Schéma simplifié (architecture distribuée sous entendue) du patron de conception MVC model 2.

7.1.1.2 MVC model 2

Dans un contexte multi utilisateurs et orienté n-tiers (par exemple un serveur Web), la mise à jour des vues concernées par une modification du modèle pose problème : le modèle n'est pas forcément connecté à la vue. D'autre part, MVC Reenskaug peut s'avérer lourd à mettre en place à cause, entre autres, de la multitude de contrôleurs à implémenter. MVC model 2 (Sun-Microsystems, figure 7.2) simplifie en autorisant le regroupement de contrôleurs et élimine la dépendance entre la vue et le modèle : le contrôleur est chargé de la notification des éventuelles modifications du modèle aux vues. Dans un contexte client-serveur, la vue est côté client, le contrôleur et le modèle sont du côté serveur, le contrôleur étant le point d'entrée du serveur.

7.1.2 Application à Umodelis

Un MVC model 2 a été utilisé pour organiser la couche présentation. D'une part, le modèle peut être local, présent dans un module de traitements du client Umodelis mais il peut aussi bien être distant comme c'est le cas pour le modèle du module de simulation : on considère que les modèles hydrologiques mis à disposition sur le serveur de modélisation sont les traitements aux requêtes de simulation. Le modèle étant distribuable, une dépendance entre la vue et le modèle ajouterait une gestion d'une connexion supplémentaire qui est évitée en utilisant MVC model 2. D'autre part, les vues d'Umodelis étant de natures différentes et de fonctionnalités différentes, il est préférable de concevoir un contrôleur par type de vue : un pour les SIG et un pour les clients Web. Le modèle, qui propose tous les traitements, n'est pas influencé par le type de la vue. La figure 7.3 schématise l'intégration de la vue d'un module de traitements dans un SIG :

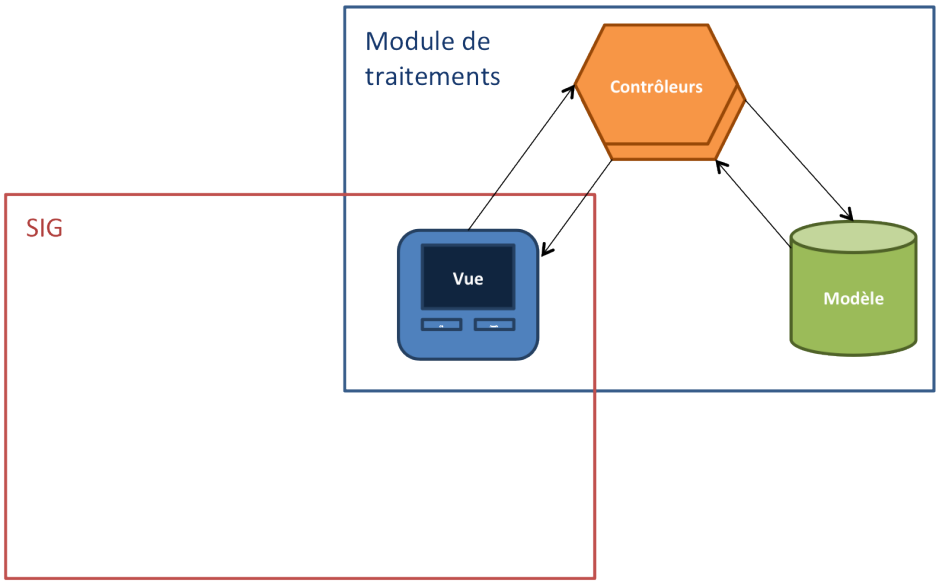

Fig. 7.3 : Schéma d'intégration d'un module de traitements dans un SIG.

La vue, en plus de son rôle d'interface utilisateur, constitue le point d'extension d'un SIG et réalise ainsi l'intégration du module de traitements dans le SIG. La vue peut éventuellement hériter d'une classe générique ou réaliser une interface, fournie par le client SIG qui spécifie le comportement du point d'extension. Il en va de même pour le cas du client Web où la vue est gérée par une application Web dynamique au sein d'un serveur Web. Il en résulte que seule la vue est dépendante du client choisi. La prochaine section

démontrera l'indépendance des contrôleurs vis-à-vis du client choisi (mais pas de son type) et le prochain chapitre celle du modèle grâce à la bibliothèque Umodelis.

7.2 Modélisation

Le couplage faible promis par MVC est obtenu par la création d'interfaces UML, abstraction des comportements de la vue, des contrôleurs et du modèle.
Les diagrammes UML qui viennent illustrer le mémoire, sont volontairement simplifiés afin de préserver leur lisibilité : les attributs de classes et les constructeurs sont cachés, seules les méthodes de classes pertinentes sont montrées sans leurs arguments ni leur type de retour et elles sont systématiquement cachées lorsqu'elles sont héritées d'une classe ou spécifiées dans une interface du diagramme.
La figure 7.4 propose le diagramme de classes UML pour la vue et les contrôleurs du module de création de sites hydrologiques : un couple pour le client uDig et un autre pour une servlet sous Apache Tomcat.

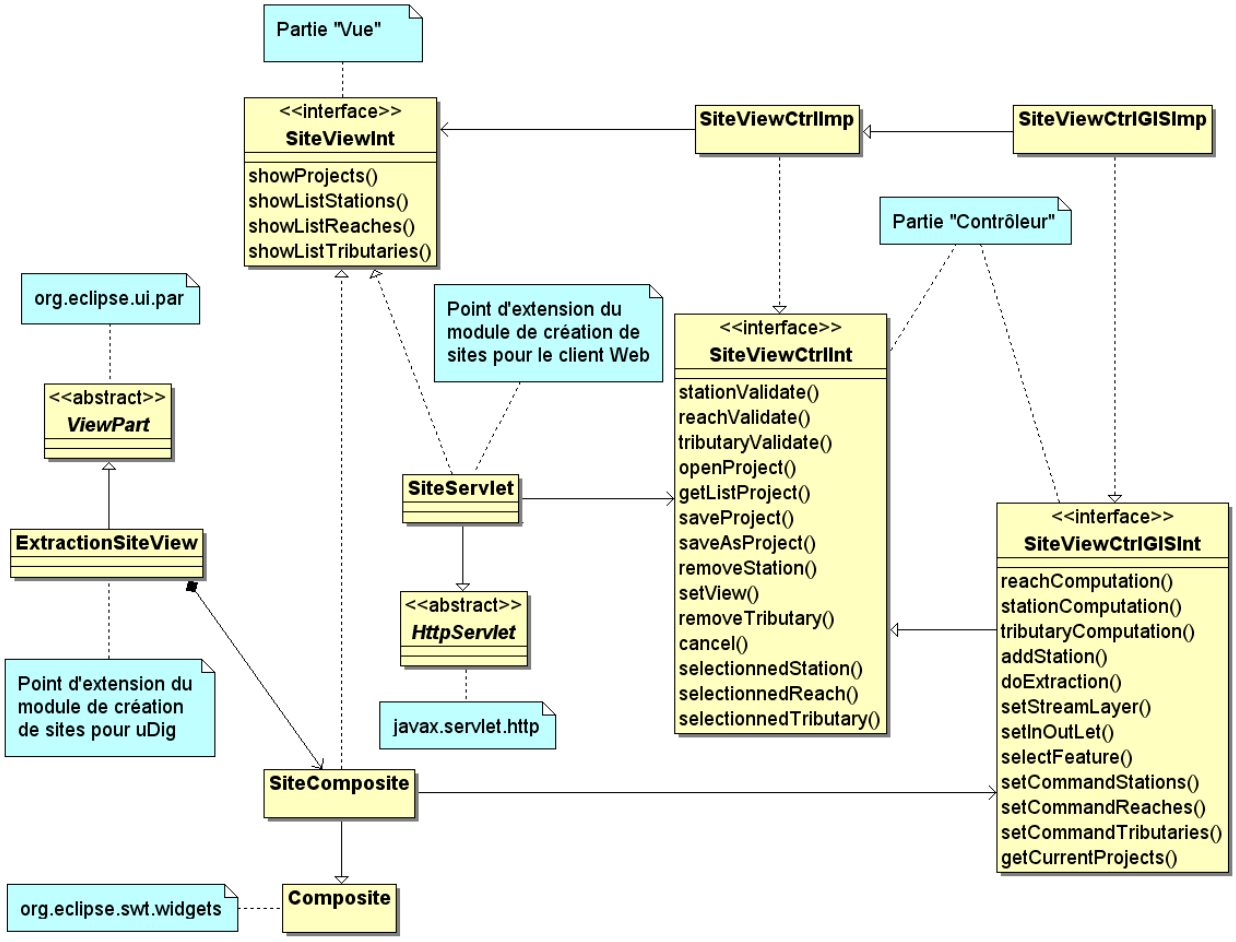

FIG. 7.4 : Diagramme de classes modélisant l'intégration du module de création de sites hydrologiques dans uDig.

L'interface SiteViewInt spécifie le comportement des vues comme par exemple l'affichage des propriétés

alphanumériques (objets hydrologiques) des sites hydrologiques (et non leur représentation graphique qui est gérée par la bibliothèque SIG présentée au chapitre 8 page 47) et des projets. Les classes concrètes, qui réalisent SiteViewInt, sont : SiteComposite qui est la vue spécifique à uDig et SiteServlet, celle spécifique au client Web. SiteComposite représente un panneau graphique qui hérite de la classe Composite de la bibliothèque SWT et accueille les contrôles SWT (boutons, etc.) nécessaires aux interactions homme-machine. Il est encapsulé dans la classe ExtractionSiteView qui hérite de ViewPart, le point d'extension abstrait d'uDig/Eclipse RCP. SiteServlet est une classe héritant du patron de servlet HttpServlet fourni par Apache/Tomcat afin de générer dynamiquement les pages Web chargées des interactions avec l'utilisateur. Les points d'entrés des clients, ExtractionSiteView pour uDig et SiteServlet pour le client Web, sont chargés de l'injection de dépendances dans leur contrôleur au moment de leur instanciation.

Comme précédemment dit, Umodelis prévoit un contrôleur par type de vue (et non pas un contrôleur par vue). SiteViewCtrlInt est l'interface du contrôleur associé à une vue comprenant les fonctions de gestion des sites hydrologiques du module sans la possibilité de création ni du rendu graphique des sites. Elle correspond aux fonctionnalités du client Web. L'analyse a démontré que le contrôleur d'un client SIG n'est qu'une augmentation des fonctionnalités du contrôleur du client Web. Ainsi l'interface et la classe concrète du contrôleur du SIG, respectivement SiteViewCtrlGISInt et SiteViewCtrlGISImp, héritent respectivement de SiteViewCtrlInt, l'interface du contrôleur du client Web et SiteViewCtrlImp, sa classe concrète. On obtient une factorisation de la gestion des sites hydrologiques, facilitant la maintenance. On obtient bien un couplage faible car les classes concrètes des contrôleurs et des vues ne font référence qu'aux interfaces du patron MVC.

7.3 Conclusion

Nous avons vu comment le client Web et le SIG sont intégrés à l'architecture d'Umodelis par l'application du patron de conception MVC model 2. Le chapitre suivant a pour objet la modélisation des structures de données des sites hydrologiques grâce à la conception de la bibliothèque Umodelis.

CHAPITRE 8

Bibliothèque Umodelis

Evoquée à la fois comme collection de structures de données ou bibliothèque graphique et d'interactions graphiques avec le client SIG, la bibliothèque Umodelis représente une autre clef, avec le patron MVC, du couplage faible avec les clients Umodelis. Ce chapitre commence par aborder la dualité de la représentation des sites hydrologiques afin d'introduire la conception de deux sous bibliothèques : la bibliothèque SIG, qui structure les objets géographique, et la bibliothèque objets hydrologiques, qui structure les données hydrologiques.

8.1 Dualité des sites hydrologiques

Un site hydrologique associe un objet géographique comme par exemple un cours d'eau et des données hydrologiques spatialisées comme l'aire drainée par le cours d'eau ou des données hydrologiques non spatialisées comme la longueur du cours d'eau. Il peut donc être représenté, soit graphiquement par son objet géographique, soit alphanumériquement par ses données associées : les objets hydrologiques. Il existe donc une dualité concernant la représentation d'un site. Le client Web d'Umodelis ne représente que les objets hydrologiques d'un site car il ne possède pas pour l'instant de couche SIG pour afficher les objets géographiques au contraire du client SIG qui exploite les deux facettes. Cette section présente mon étude sur la modélisation informatique des concepts des objets géographiques commune aux SIG les plus distribués puis se termine par une synthèse sur la conception de la bibliothèque Umodelis que j'ai proposé puis implémenté.

8.1.1 Objets géographiques

De l'étude sur la représentation des objets géographiques par les SIG les plus répandus (ArcGIS, OpenJump et Geotools utilisé par uDig), j'ai déduit la définition de concepts communs suivants :

- Feature :

Une feature représente la structure générique des objets géographiques. Cette structure se compose de propriétés (données spatialisées) et d'une figure géométrique ou géométrie. Elle comprend au moins une propriété qui joue le rôle d'identifiant unique au sein d'une layer (collection de features).

- Geometry :

C'est une structure générique qui renferme les informations sur la forme géométrique de l'objet géographique : un ensemble de coordonnées avec une équation d'interpolation pour la représentation graphique de l'espace entre les coordonnées.

- FeatureType ou modèle de features :

Le featureType est une structure générique qui rassemble les métadonnées des propriétés des features de même nature : il reflète la nature de l'objet géographique. Il comporte, entre autres, le type de forme géométrique des features (point, line string, polygon, etc.). C'est une matrice servant à fabriquer des features.

- Layer :

C'est une collection de features de même nature (même featureType). Elle sert à l'organisation des features et en offre une représentation graphique en couche superposable. Distribuable, elle est identifiée par son Uniform Ressource Locator (URL). Elle se charge également d'appliquer le système de projection de coordonnées choisi par l'utilisateur.

- Map :

Une map est un assemblage ordonné de layers selon le principe que les features d'une layer d'ordre supérieur occultent celles des layers d'ordre inférieur. C'est le container de plus haut niveau de l'information spatiale.

- Filter :

Filter est un système de sélection multicritères de features au sein d'une layer. Il existe deux types de filter : filter de propriétés et filter géométrique. Le filter de propriétés discimine les features selon la valeur de leurs propriétés (identifiant, etc.). Le filter géométrique discrimine les features selon la relation entre leur géométrie et celle donnée (intersection, recouvrement, etc.).

8.1.2 Synthèse

L'un des buts du projet Umodelis est de faciliter le formatage des données en entrée des modèles hydrologiques. Sa réalisation passe par l'association des structures de représentations graphiques venant des SIG avec les structures de données hydrologiques. En termes d'organisation logicielle, deux bibliothèques distinctes ont été créées, la bibliothèque SIG et la bibliothèque objets hydrologiques, pour les raisons suivantes :

8.1.2.1 Encapsulation des dépendances SIG

Le souci permanent de la conception d'Umodelis est de maitriser la dépendance d'Umodelis vis-à-vis du client SIG. Regrouper toutes les dépendances SIG dans une bibliothèque à part participe à leur encapsulation. Ainsi la bibliothèque SIG regroupe toutes les structures de représentation des objets géographiques ainsi que les fonctions d'interactions avec le SIG.

8.1.2.2 Utilisation séparée

Compte tenu de l'absence de couche graphique SIG dans le client Web et de son unique gestion de l'aspect alphanumérique des sites hydrologiques, la séparation des sémantiques des deux bibliothèques apparaît pertinente : utiliser les objets hydrologiques indépendamment d'une implémentation de bibliothèque d'objets géographiques liée à un SIG. Néanmoins, cette séparation ne va pas sans poser des problèmes comme la cohérence de la sauvegarde des sites hydrologiques. Umodelis doit se reposer au maximum sur les fonctions avancées proposées par le SIG.
Le cas de la sauvegarde d'objets géographiques en fait partie. On se retrouve donc avec l'aspect graphique des sites hydrologiques enregistré séparément de sa description alphanumérique. La bibliothèque objets hydrologiques rassemble les structures des concepts hydrologiques dont la persistance est à la charge du serveur de modélisation. Ces structures seront chargées d'établir un lien (éventuellement inexistant) avec les structures de représentation des objets géographiques. Bien que les features possèdent des propriétés

pouvant être utilisées pour associer les objets géographiques avec les données non spatialisées, Umodelis ne s'en sert pas afin d'éviter la redondance de l'information : la bibliothèque SIG est uniquement en rapport avec l'aspect graphique du site et la bibliothèque objets hydrologiques uniquement en rapport avec les données alphanumériques hydrologiques.

8.2 Bibliothèque SIG

Le but de cette partie n'est pas de présenter la bibliothèque SIG in extenso mais de montrer les principes qui ont prévalus à sa conception.

8.2.1 Encapsulation de la dépendance

L'idée d'encapsuler la dépendance entre le SIG et Umodelis a pour but de rendre ce couplage le plus faible possible. Le patron de conception adaptateur (design pattern adapter ; GAMMA *et al.*, 1994 ; voir annexe A.1) est une solution architecturale élégante afin de réaliser un tel couplage. En proposant une série d'interfaces pour les structures SIG communes évoquées ci-dessus, les modules qui les utiliseront ne seront pas dépendants du choix du SIG. Seules les implémentations de ces interfaces seront liées à la bibliothèque du SIG choisi.
Voici deux exemples d'encapsulation de structures SIG qui illustrent bien le principe de ce patron de conception appliqué pour la bibliothèque SIG : les classes des formes géométriques et la classe des fonctions d'interactions SIG.

8.2.1.1 Formes géométriques

L'analyse des formes géométriques (figure 8.1) a conduit à la création d'une super interface GeometryInt qui regroupe toutes les spécifications communes aux formes géométriques : accesseurs aux propriétés de la forme (nombre de coordonnées, type, etc.) et opérations géométriques (intersection, recouvrement, etc.). Les interfaces filles de GeometryInt spécialisent la forme : LineStringInt est l'interface des lignes par interpolation linéaire de coordonnées et PointGeometryInt représente des points. La classe GeometryUdigImp implémente les méthodes de l'interface GeometryInt spécifiquement pour le client uDig en s'appuyant sur son interface de programmation (Application Programming Interface ou API). Par soucis de cohérence et afin de factoriser un maximum de code, les classes spécifiques à uDig réalisant les interfaces filles de GeometryInt héritent de GeometryUdigImp.
Le diagramme de la figure 8.2 illustre l'application du patron de conception adaptateur pour les formes géométriques. GeometryInt est l'interface de l'adaptateur que les modules de traitements utiliseront, GeometryUdigImp, l'adaptateur concret, est une implémentation de GeometryInt spécifique à uDig par composition d'un objet de type Geometry (bibliothèque Geotools/JUMP), la classe adaptée. Les méthodes de GeometryUdigImp sont autant de relais vers celles sémantiquement proches de l'objet Geometry encapsulé, l'implémentation réelle de la géométrie. L'injection de dépendance par construction explique le lien de composition entre l'adaptateur et l'adapté : un objet de type Geometry est donné en paramètre du constructeur de GeometryUdigImp au moment de son instanciation.

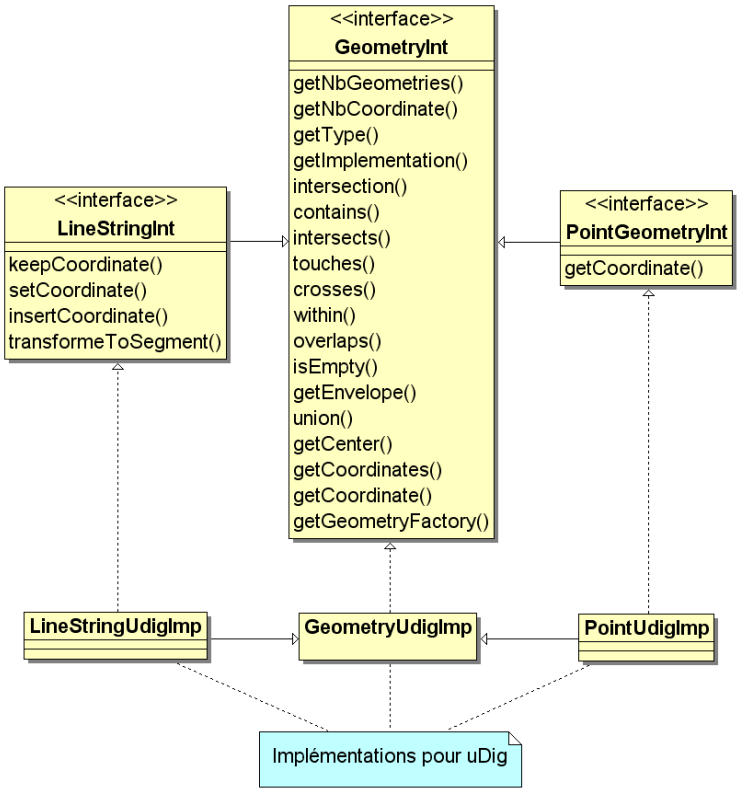

FIG. 8.1 : Extrait du diagramme de classes des interfaces des formes géométriques.

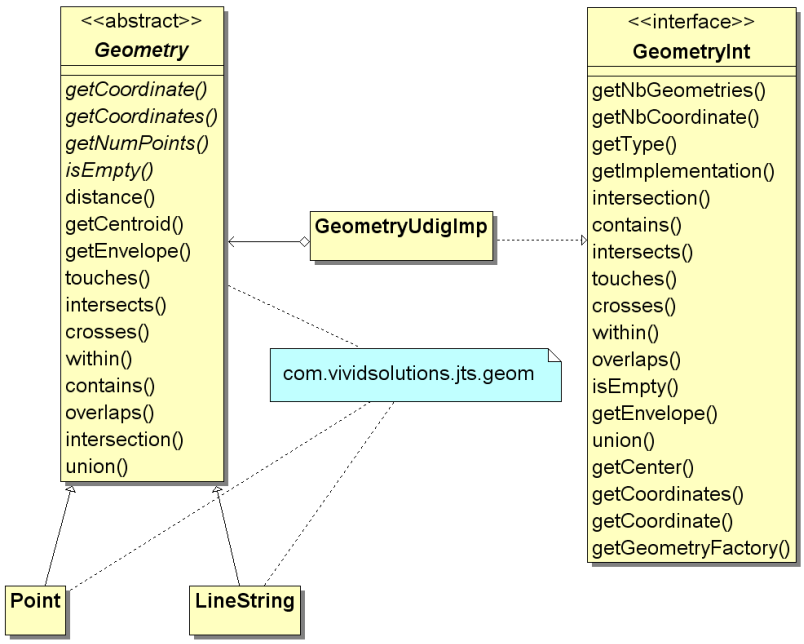

FIG. 8.2 : Extrait du diagramme de classes de l'adaptation des classes géométries de l'API d'uDig.

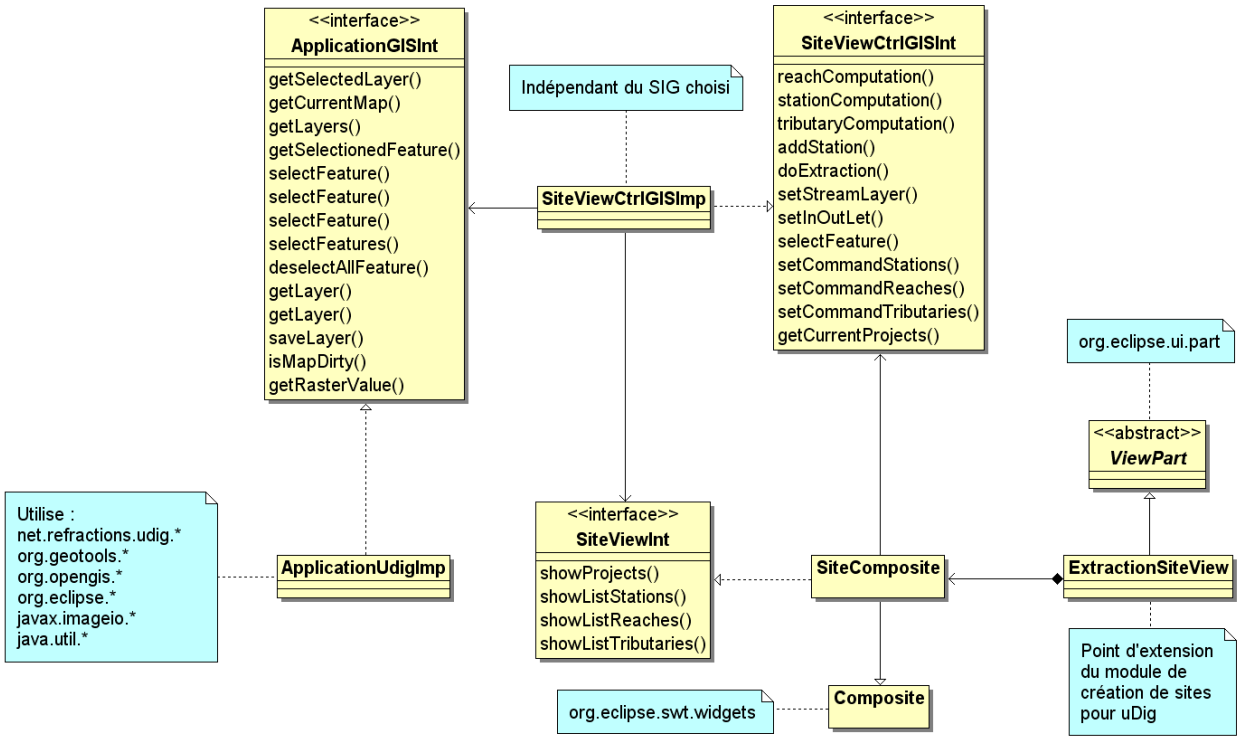

FIG. 8.3 : Extrait du diagramme de classes de l'encapsulation des fonctions SIG.

8.2.1.2 Fonctions d'interactions SIG

La figure 8.3 est un autre exemple de cette encapsulation de dépendance par le patron de conception adaptateur et illustre également son utilisation au sein du patron MVC détaillé au chapitre précédent. SiteViewCtrlGISImp est le contrôleur concret du module de création de sites hydrologiques dont le comportement est spécifié par son interface SiteViewCtrlGISInt. Le contrôleur concret fait appel aux fonctions d'interactions SIG de l'interface ApplicationGISInt dont la classe ApplicationUdigImp en est une réalisation pour uDig. En ne faisant référence qu'à des interfaces, le contrôleur est faiblement couplé avec le client SIG choisi. Comme indiqué à la section 7.2 du chapitre 7, l'injection de dépendance s'effectue par construction : pour cet exemple le point d'extension d'uDig, ExtractionSiteView se charge de passer un objet ApplicationUdigImp au contrôleur SiteViewCtrlGISImp au moment de instanciation de ce dernier.

Si le patron de conception adaptateur réalise l'encapsulation de la spécificité de la vue par des interfaces, la prochaine section traite de la création d'objets de cette bibliothèque sans dépendance directe.

8.2.2 Instanciation découplée d'objets

La création d'objets de la bibliothèque SIG implique la connaissance du type concret de celui-ci. Or une telle connaissance revient à créer une dépendance vis-à-vis du client SIG retenu. Cette dépendance peut être maitrisée par l'application du patron de conception fabrique abstraite (design pattern abstract factory ; GAMMA *et al.*, 1994 ; voir annexe A.6). La figure 8.4 illustre l'application de ce patron à propos de la

création de layers :

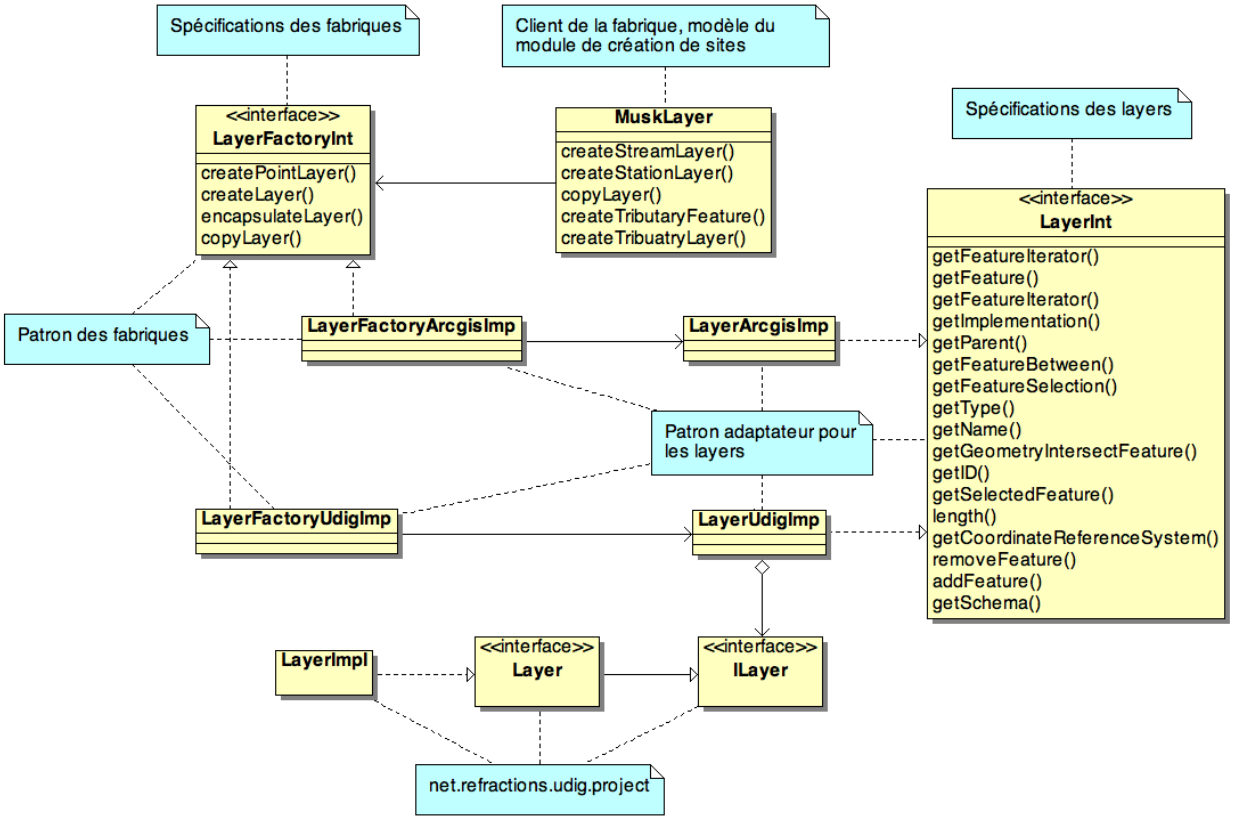

FIG. 8.4 : Extrait du diagramme de classes de fabriques de layers.

L'idée de ce patron de conception est de déléguer la construction d'objets définis par une interface à un intermédiaire qui encapsule la dépendance au type concret de ces objets : la fabrique.
Dans cet exemple apparait le patron de conception adaptateur formé par LayerInt, l'interface des objets layers d'Umodelis et ses implémentations : LayerArcgisImp et LayerUdigImp qui sont chargées d'encapsuler les objets layers provenant des bibliothèques des clients SIG pour lesquels elles sont spécifiques : respectivement ArcGIS (non représentée) et uDig (LayerImpl).
L'instanciation des objets des classes concrètes de LayerInt est à la charge de fabriques spécifiques au client SIG et dont le comportement est décrit par l'interface LayerFactoryInt. Le découplage est réalisé lorsque le prescripteur de création de layers par exemple la classe MuskLayer ne fait référence qu'aux méthodes de l'interface des fabriques. Comme toujours, c'est l'injection de dépendance par construction qui permet à MuskLayer d'obtenir un objet implémentant une fabrique de layers.
Le point suivant concerne l'aspect programmation événementielle des clients SIG, une approche originale est proposée : le relais d'observations.

8.2.3 Relais d'observations

Observateur est le patron de conception (design pattern observer ; GAMMA *et al.*, 1994 ; voir annexe A.8) couramment utilisé pour la gestion d'événements et celui qui a été choisi pour Umodelis. La figure 8.5

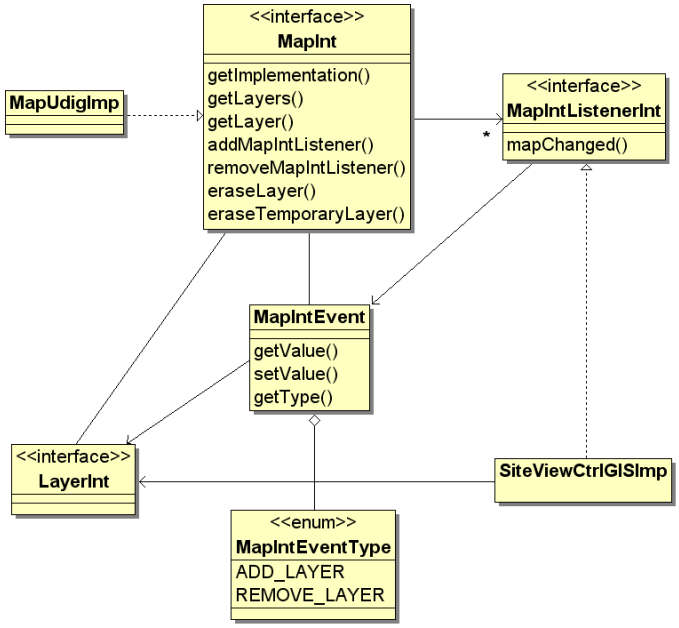

FIG. 8.5 : Extrait du diagramme de classes du mécanisme d'observation de cartes d'Umodelis.

montre une utilisation de ce patron pour la gestion des événements de modification des couches contenues dans une carte.
Le principe est basé sur l'abonnement d'entités observatrices aux notifications déclenchées par une entité observée. Les observateurs implémentent une interface d'abonnés qui spécifie une méthode de traitements de l'événement et l'entité observée propose un mécanisme d'abonnement. Lors de l'occurrence d'un événement, l'entité observée exécute la méthode de traitements d'événements de ses abonnés.
Dans notre cas, le contrôleur de type client SIG est informé des modifications de l'utilisateur sur la carte courante. Pour cela, l'entité observatrice, SiteViewCtrlImp, le contrôleur, implémente la méthode mapChanged spécifiée dans l'interface MapIntListenerInt et s'abonne aux événements de modifications des layers à l'aide de la méthode addMapIntListener proposée par MapUdigImp, classe d'encapsulation des objets map d'uDig (non représentés). Les événements sont symbolisés par la classe MapIntEvent qui contient la nature de l'événement (MapIntEventType) et l'objet de type LayerInt concerné par la modification.
La gestion des événements d'uDig utilisant le même patron de conception, un mécanisme d'interception et de relais d'événements entre les deux systèmes est mis en place. La figure 8.6 illustre le relais entre la gestion des événements de map d'uDig à celle d'Umodelis évoquée plus haut. La classe MapUdigImp joue le rôle du point de jonction entre les deux mécanismes : elle est un observateur des événements déclenchés par l'objet MapImpl d'uDig qu'elle encapsule, en implémentant l'interface d'abonnés IMapCompositionListener et relaie les événements map uDig en événements map Umodelis qu'elle propage à ses abonnés (SiteViewCtrlGISImp).

C'est à l'aide de ces trois patrons de conception que l'indépendance du contrôleur et du modèle des modules de traitements d'Umodelis vis-à-vis du client SIG est assurée.

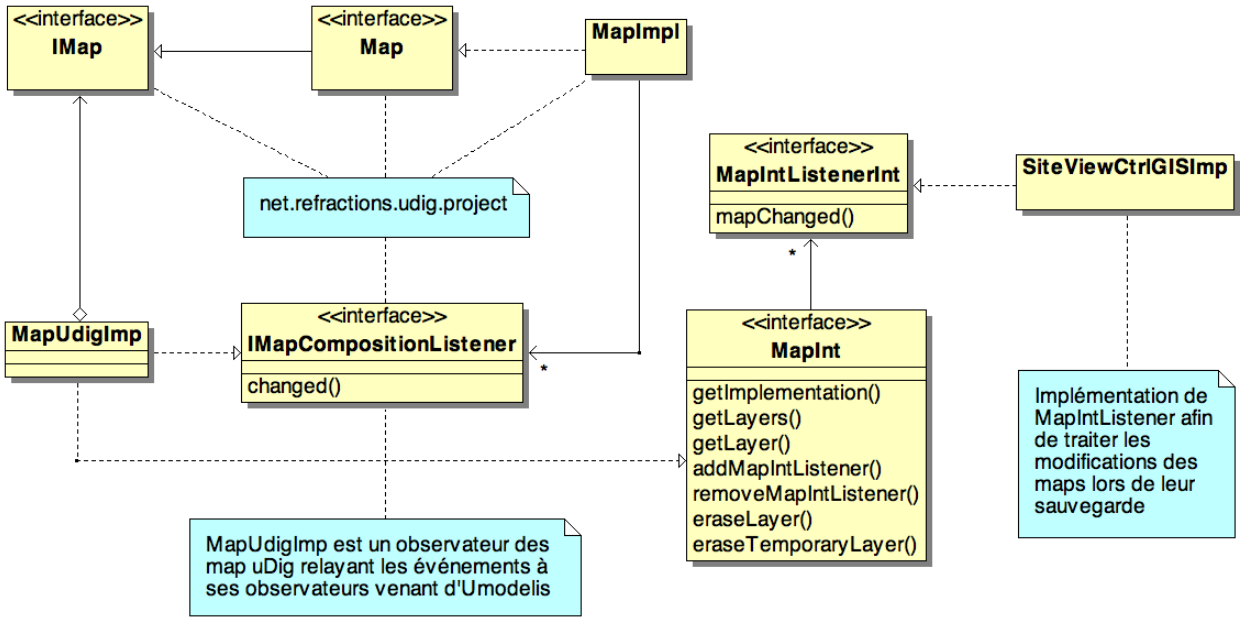

FIG. 8.6 : Extrait du diagramme de classes du mécanisme d'observation de map d'uDig relayé à celui d'Umodelis.

8.3 Bibliothèque objets hydrologiques

Les structures de données qui forment la bibliothèque objets hydrologiques sont inspirées des bases de données de l'IRD. Les concepts géomatiques étant utilisés depuis de nombreuses années par l'IRD, j'ai étudié leur représentation en base afin d'en apporter des modifications mineures pour leur utilisation au sein d'Umodelis. La présentation de cette étude se limite volontairement aux stations, reaches et tributaries, les autres entités comme cotes (hauteurs d'eau), débits et capteurs ne sont pas représentées car elles sont indirectement associées aux stations et aux tributaries et ne sont pas concernées par les modifications.

8.3.1 Modélisation de la base de données Muskingum

La figure 8.7 présente un extrait du Modèle Conceptuel de Données (MCD) de la base de données Muskingum de l'IRD (PostgreSQL). Deux stations délimitent un reach et un reach possède éventuellement des tributaries de façon exclusive. Cependant, il manque un sur-ensemble aux stations et aux reaches (et leurs tributaries) mis bout à bout afin de les rassembler pour former un cours d'eau et ses stations.

8.3.2 Modifications

C'est dans ce but que l'entité projects est créée (figure 8.8). Un project possède au moins un reach et deux stations qui délimitent le reach. Notion évoquée à la section 3.4 page 15, le project assure une cohérence à une étude hydrologique formée de stations hydrométriques et de portions de cours d'eau. De plus les paramètres utiles aux modèles hydrologiques y sont incorporés.
Enfin, l'introduction de l'attribut layer_url stockant une URL dans chacune des entités (sauf projects) ap-

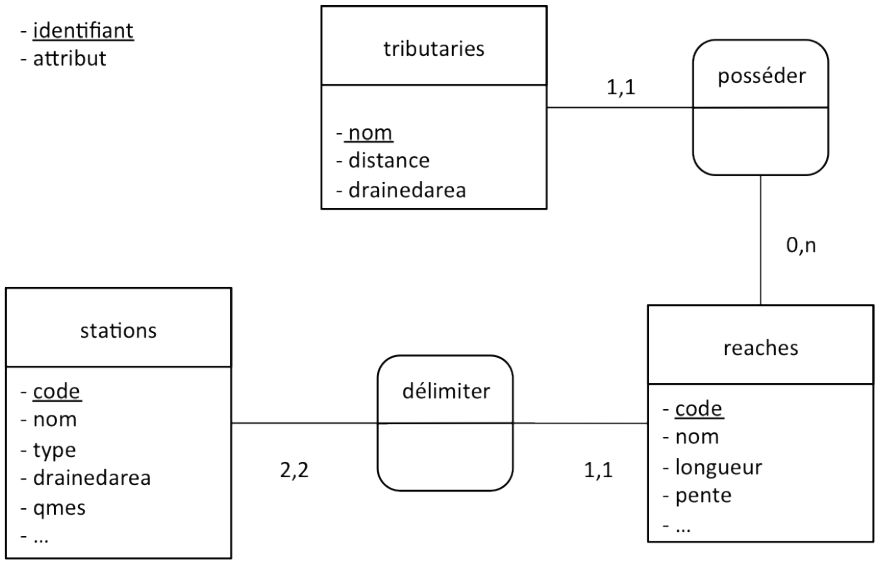

FIG. 8.7 : Extrait du MCD de la base de données Muskingum.

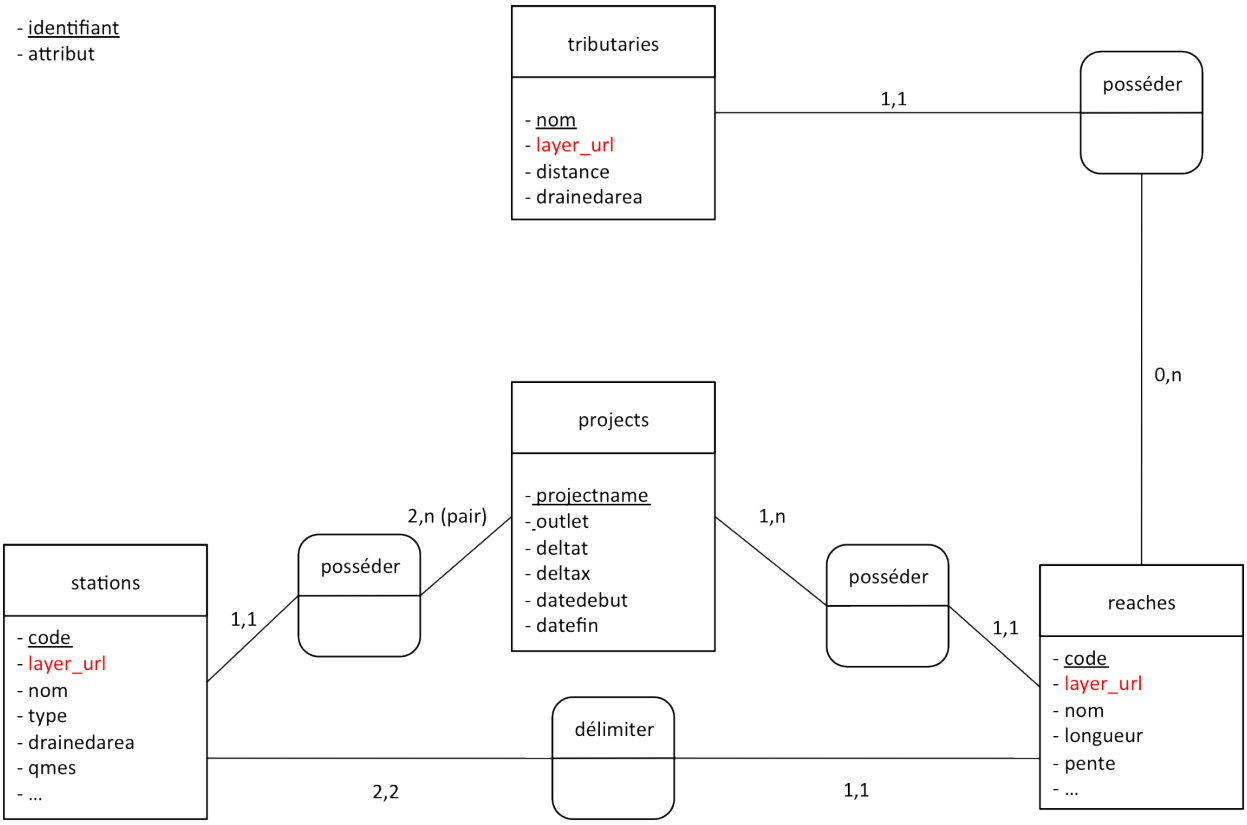

FIG. 8.8 : Extrait du MCD de la base de données Muskingum modifiée.

porte une solution pour lier la représentation alphanumérique des sites hydrologiques avec leur représentation graphique : les features qui représentent les objets géographiques, sont contenues dans des layers identifiables et localisables par leur URL. En faisant correspondre d'une part le couple attribut code et identifiant de feature et d'autre part le couple layer_url et URL de layer, nous arrivons à établir le lien entre objets hydrologiques et objets géographiques.

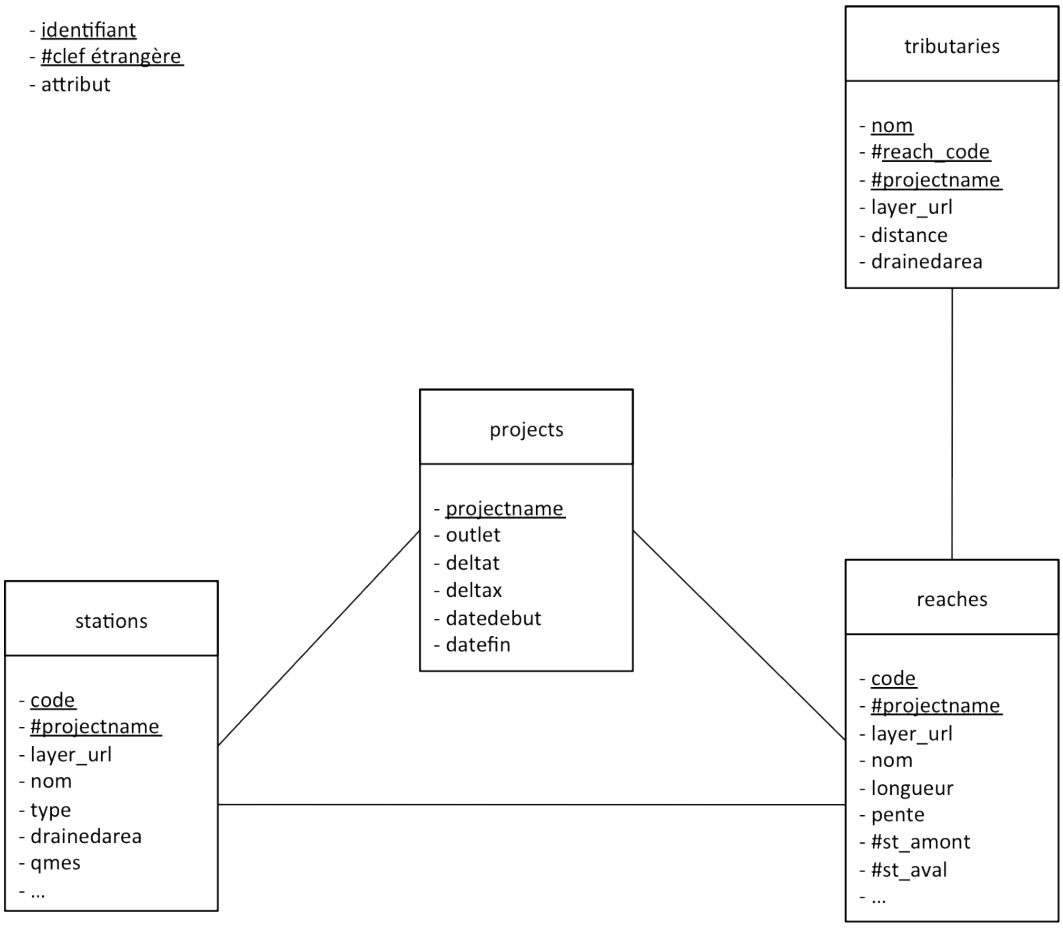

Fig. 8.9 : Extrait du MLD de la base de données Muskingum modifiée.

La figure 8.9 donne le Modèle Logique de Données (MLD) après application des règles de transformation d'un MCD. Les relations binaires de type 1 : n ou 2 : n disparaissent au profit d'une clef étrangère introduite dans l'entité de cardinalité 1,1 qui fait référence à la clef primaire de l'entité de cardinalité 1 : n ou 2 : n. L'attribut projectname, l'identifiant de la table projects, devient donc une clef étrangère dans les tables stations et reaches.

Toutefois, nous devons prendre en compte que les modèles hydrologiques utilisent les colonnes des stations, des reaches ou des tributaries comme variables : une station ou un reach ne peut pas appartenir à un autre project. Le code des stations et des reaches n'est donc plus suffisant pour assurer leur identification. Cette clef étrangère doit participer à la formation de leur clef primaire en association avec la colonne code. Il en va de même pour la relation entre reaches et tributaries qui hérite des clefs étrangères reach_code et projectname. Concernant la relation 2 :1 qui traduit la définition d'un reach par deux stations à ses

extrémités, elle disparait au profit de l'introduction des codes des deux stations en clef étrangères dans la table reaches : st_amont signifiant station en amont et st_aval, station en aval.

8.3.3 Traduction en classes UML

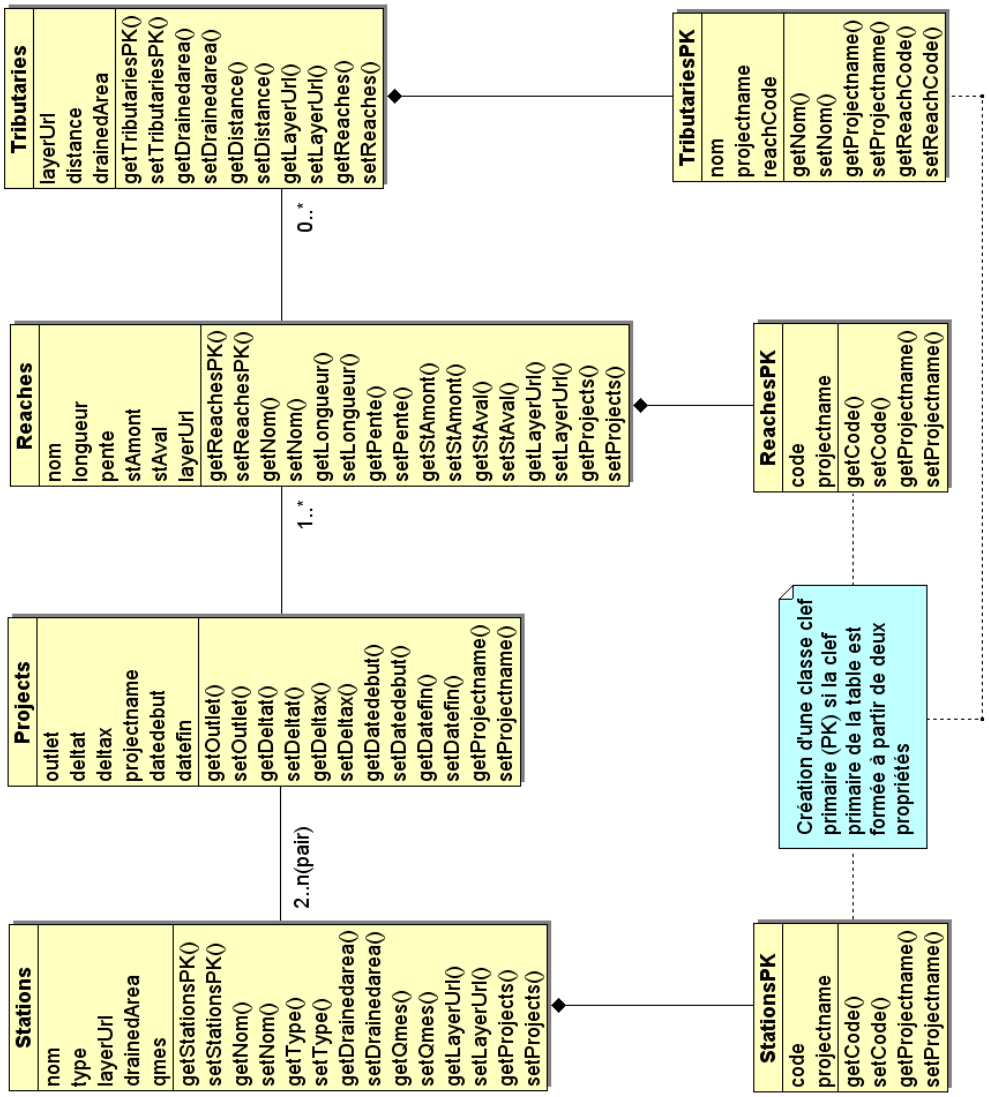

FIG. 8.10 : Traduction de l'extrait du MLD de la base de données Muskingum en classes UML.

La bibliothèque objets hydrologiques étant à l'intention de Java, un langage d'implémentation objet, il est nécessaire de traduire les tables de la base de données en classes UML. La persistance des instances de ces classes en base de données sera traitée au cours du chapitre 11.
La traduction du MLD en diagramme de classes UML est relativement simple : les tables sont traduites en classes et leurs colonnes, hormis celles qui participent à une clef primaire composée, sont traduites en attributs de classe de type équivalent (figure 8.10).

La composition de colonnes pour former une clef primaire n'a pas d'équivalent sémantique dans la notation UML : plusieurs attributs de classes n'expriment pas la cohérence d'une clef primaire, il faut donc les encapsuler dans une classe qui joue le rôle de clef primaire et dont le nom est formé à partir de celui de la table et du suffixe PK (Primary Key). Pour le cas de la table projects, une seule colonne forme la clef étrangère, projectname de la classe Projects est donc l'attribut identifiant. Pour la table stations où code et projectname forment la clef primaire, la traduction en UML passe par la création de la classe StationsPK ayant pour attributs code et projectname et compose la classe Stations.
Enfin, tous les attributs de classe sont accessibles publiquement via des méthodes accesseurs (setters et getters) dont le nom est formé à partir du préfix set ou get et du nom de l'attribut.

8.4 Conclusion

La bibliothèque Umodelis est un élément fondamental de la plateforme. Elle assure d'une part un couplage faible entre les modules de traitements et le client SIG par sa bibliothèque SIG et d'autre part, elle permet la conception de clients sans couche SIG en séparant la représentation graphique des sites hydrologiques de leurs données alphanumériques par l'intermédiaire de la bibliothèque objets hydrologiques. Le chapitre suivant aura pour thème la conception du module de création de sites hydrologiques.

CHAPITRE 9

Module de création de sites hydrologiques

Après un bref rappel sur les cas d'utilisation du module de création de sites hydrologiques, ce chapitre traitera les points les plus intéressants de sa conception dont j'ai eu la responsabilité ainsi que celle de son implémentation. Les cas d'utilisations, les extractions d'objets géographiques, des données spatialisées et non spatialisées ainsi que la gestion des objets hydrologiques du module seront abordés.

9.1 Cas d'utilisations

Le module permet à l'utilisateur de créer un projet qui regroupe un ensemble de sites hydrologiques (voir la section 4.2.2 page 22). Dans une première approche, la création d'un projet revient à extraire des objets géographiques qui représentent un cours d'eau délimité par deux stations hydrométriques, en incluant les éventuelles stations intermédiaires situées sur le cours d'eau (figure 9.1, cas extraire des sites hydrologiques).

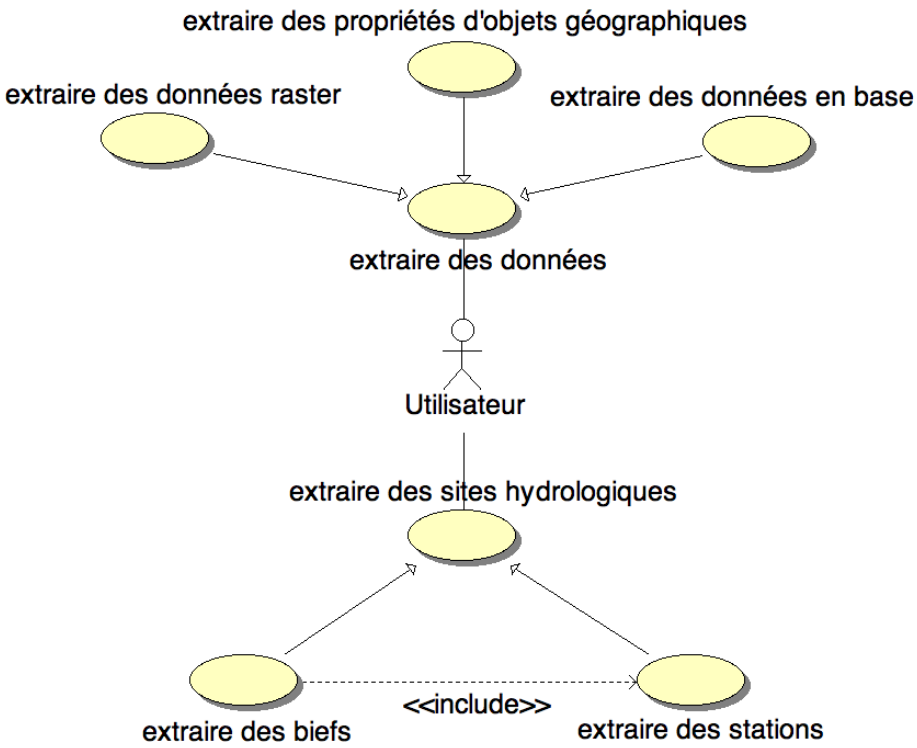

FIG. 9.1 : Diagramme de cas d'utilisation du module de création de sites hydrologiques, première partie.

Deux possibilités de paramétrer les sites hydrologiques sont offertes : soit par extractions, soit par édition manuelle. Le cas d'utilisation extraire des données regroupent l'acquisition de données provenant des rasters, des propriétés des objets géographiques contenus dans des couches ou des bases de données (figure 9.1). Le cas éditer des objets hydrologiques représente la possibilité de saisir les paramètres à l'aide d'une IHM (figure 9.2).

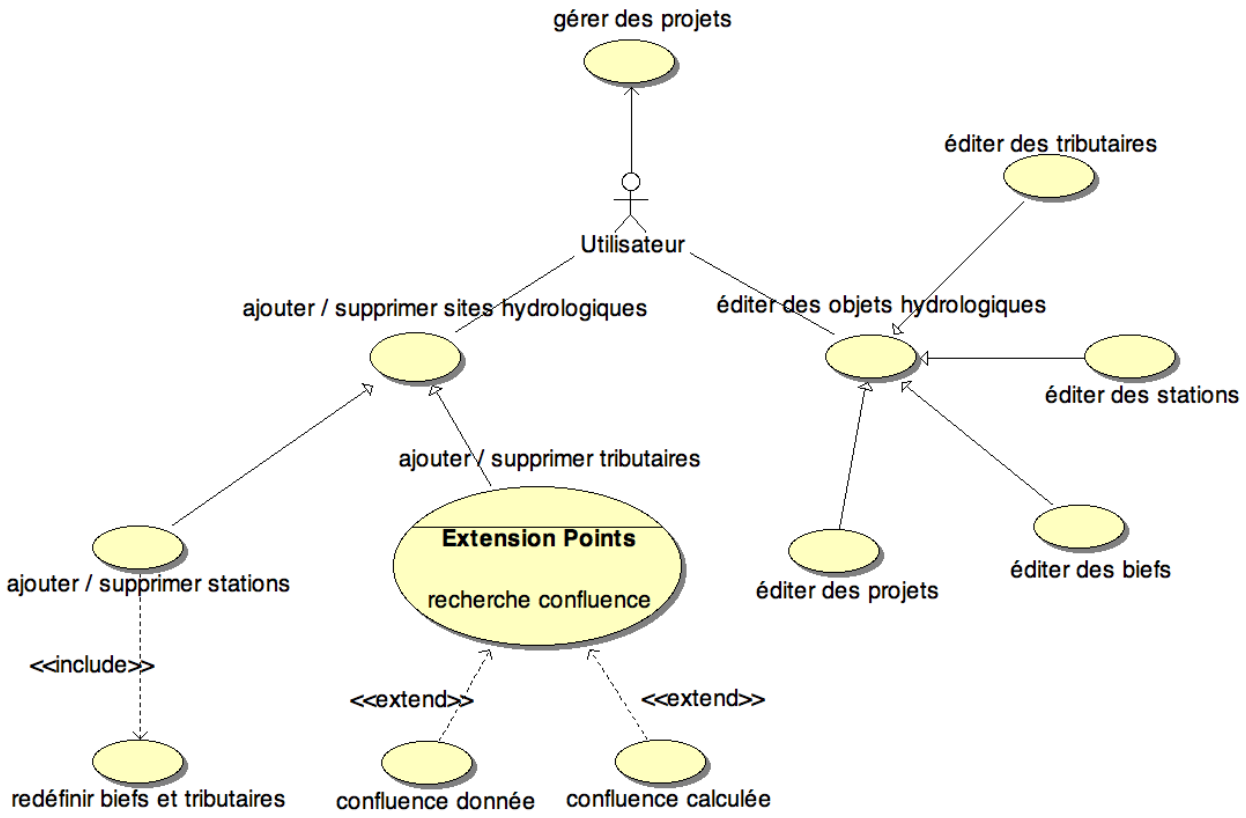

Fig. 9.2 : Diagramme de cas d'utilisation du module de création de sites hydrologiques, deuxième partie.

Concernant les autres fonctions de gestion des sites, le module permet l'ajout et la suppression de sites hydrologiques dont les stations et les tributaires, mais pas la suppression de biefs qui revient finalement à créer un nouveau projet (figure 9.2, cas ajouter/supprimer sites hydrologiques). L'ajout de tributaires possède une particularité, la confluence peut être définie par édition manuelle de la distance entre la confluence et la station en aval du bief possédant le tributaire ou elle peut être calculée par l'intersection du bief et de la feature représentant le tributaire (figure 9.2, cas ajouter/supprimer tributaires).
Enfin, le module offre également les services classiques de gestion pour les projets (figure 9.2, cas gérer projets) : la sauvegarde (persistance) et le rappel de projets ainsi que les sites associés et l'effacement de projets.

Comme l'introduction de ce chapitre le précise, seuls les points pertinents du point de vue conception seront abordés concernant les cas d'utilisation décrits précédemment. Au sujet de l'extraction de sites hydrologiques : les opérations géométriques et l'algorithme d'extraction seront présentés. L'extraction des données sera l'occasion de traiter la gestion des métadonnées des différentes sources d'information et la mise en place d'extractions programmées en lot par l'utilisateur. Enfin, seule la conception des gestion-

naires d'objets hydrologiques sera traitée, c'est-à-dire les mécanismes d'édition, d'ajout et de suppression des sites hydrologiques ainsi que ceux des projets.
L'algorithme d'ajout et suppression de sites ne présente pas d'intérêt particulier et réutilise en grande partie les opérations géométriques développées pour l'extraction des sites. Enfin, l'édition des objets hydrologiques est résolue par la conception d'une IHM exposant l'accès aux fonctions des gestionnaires d'objets hydrologiques.

9.2 Extraction des sites hydrologiques

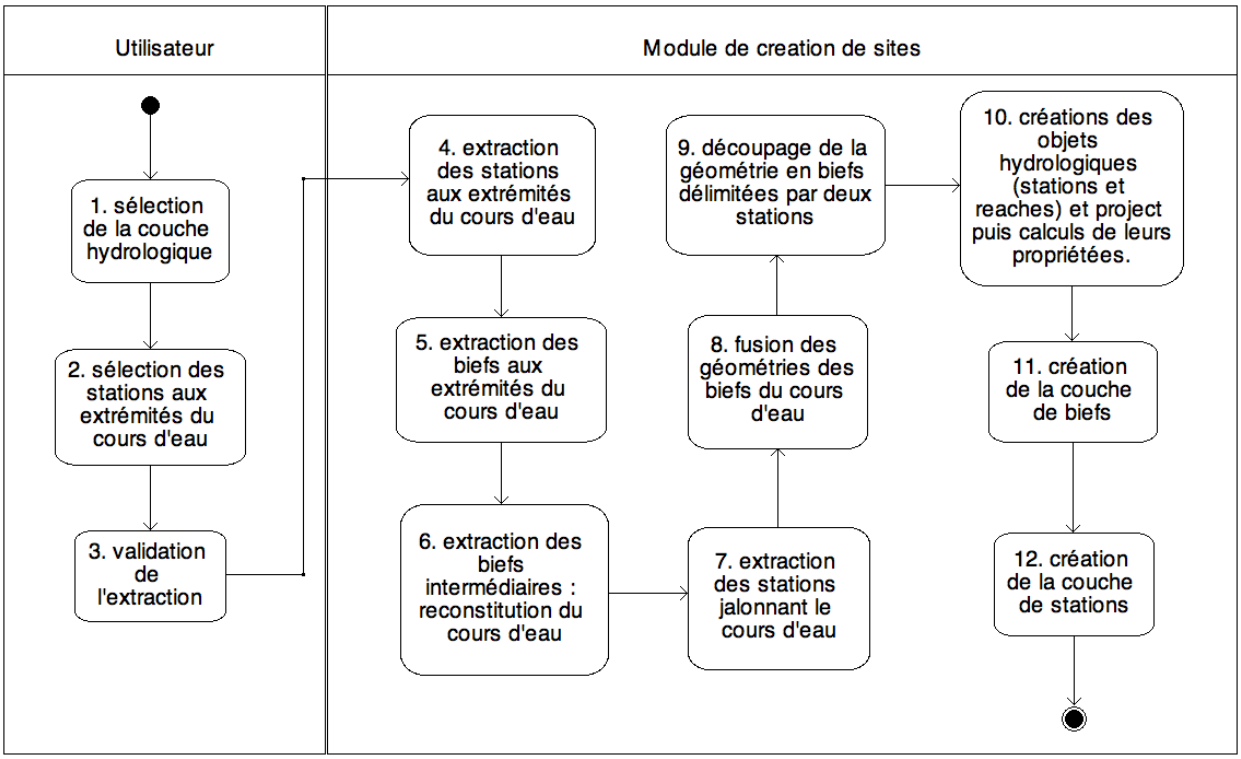

FIG. 9.3 : Diagramme d'activité de l'extraction d'un cours d'eau et de ses stations.

La figure 9.3 présente le diagramme d'activité UML de la création d'un projet par extraction de sites hydrologiques. Cette extraction consiste actuellement à extraire les features d'une couche représentant un cours d'eau délimité par deux stations elles même situées sur une couche différente. Elle s'accompagne de l'extraction des stations intermédiaires bordant le cours d'eau et la redéfinition des biefs du cours d'eau.
Afin d'illustrer le diagramme d'activité, nous allons étudier la figure 9.4 qui présente un exemple classique d'extraction : les biefs du cours d'eau sont délimités par des confluences et les couches hydrographique (biefs) et station sont décalées : les stations ne sont pas exactement sur le cours d'eau. Le décalage des couches s'explique par des simplifications introduites par la modélisation des objets géographiques (les cours d'eau n'ont pas de largeur).

Les étapes 1, 2 et 3 du diagramme d'activité 9.3, sont des étapes de préparation de la création du projet

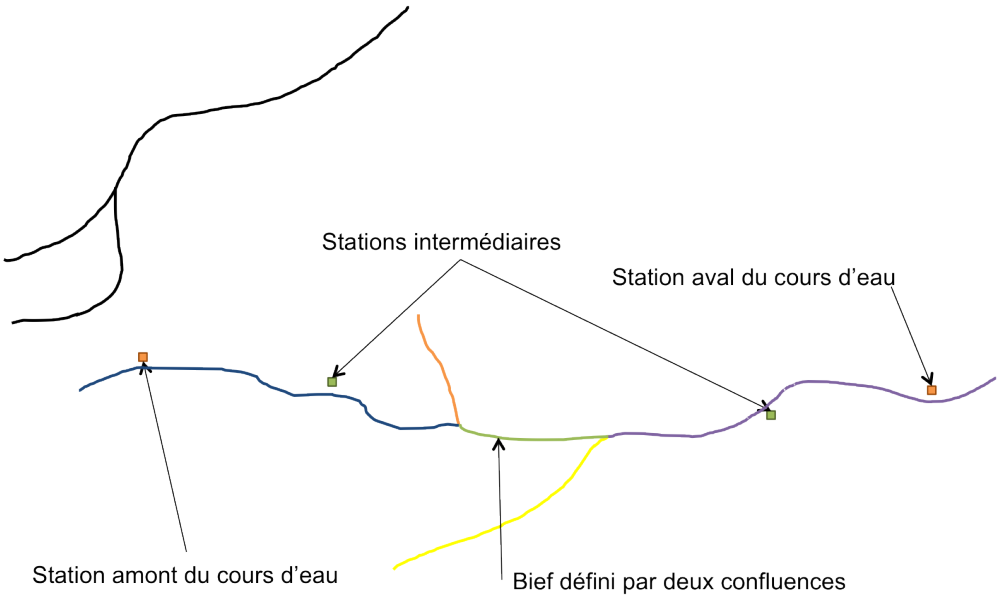

FIG. 9.4 : Schéma d'un exemple d'extraction d'un cours d'eau et de ses stations.

au cours desquelles l'utilisateur choisit la couche hydrographique qui représente le cours d'eau étudié et les deux stations qui vont délimiter la portion du cours d'eau à extraire, sur une couche représentant les stations hydrométriques.
L'étape 4 est la première activité prise en charge par le module de création de sites hydrologiques. Elle consiste à extraire les features représentant les stations choisies par l'utilisateur par application d'un filtre de propriétés sur la couche de stations (voir filter dans la section 8.1.1 page 47). L'étape 5, illustrée par la figure 9.5, a pour but de trouver le bief du cours d'eau étudié le plus proche pour chacune des stations, compte tenu du décalage des couches. La recherche s'effectue en appliquant un filtre géométrique sur la couche hydrographique qui extrait les biefs coupant une enveloppe carrée centrée sur la station. Le choix du bief est basé sur la plus courte distance entre la station et les biefs trouvés.
L'étape 6 consiste à reconstituer le cours d'eau étudié à partir des biefs déterminés lors de l'étape précédente (biefs du cours d'eau à ses extrémités) par interrogation de la couche hydrographique. A noter que l'ordre amont-aval des biefs est préservé. L'étape 7 qui consiste à extraire les stations intermédiaires, reprend le principe de l'étape 5 mais dans le sens contraire : l'enveloppe est centrée sur un bief du cours d'eau et les stations contenues par cette enveloppe sont extraites. Cependant, si l'application du filtre évite le doublonnage de stations (une station peut être contenue par les enveloppes de biefs contiguëes), elle ne garde pas l'ordre amont-aval. L'étape 7 se poursuit par l'appariement des stations avec les biefs ordonnées du cours d'eau afin de déterminer l'ordre des stations.
Les étapes 8 et 9 illustrées par la figure 9.6 ont pour but la redéfinition des biefs du cours d'eau : de la délimitation par deux confluences à la délimitation par deux stations. Cette transformation commence par la mise bout à bout des line string des features du cours d'eau afin de former une seule line string puis le découpage de celle-ci selon la projection orthogonale des coordonnées des stations.
A cause des méandres parfois tortueux des cours d'eau, plusieurs projetés orthogonaux d'une station peuvent être valables sur la line string représentant le cours d'eau (voir figure 9.7).
La solution retenue passe par la projection orthogonale de la station sur les segments de la line string en

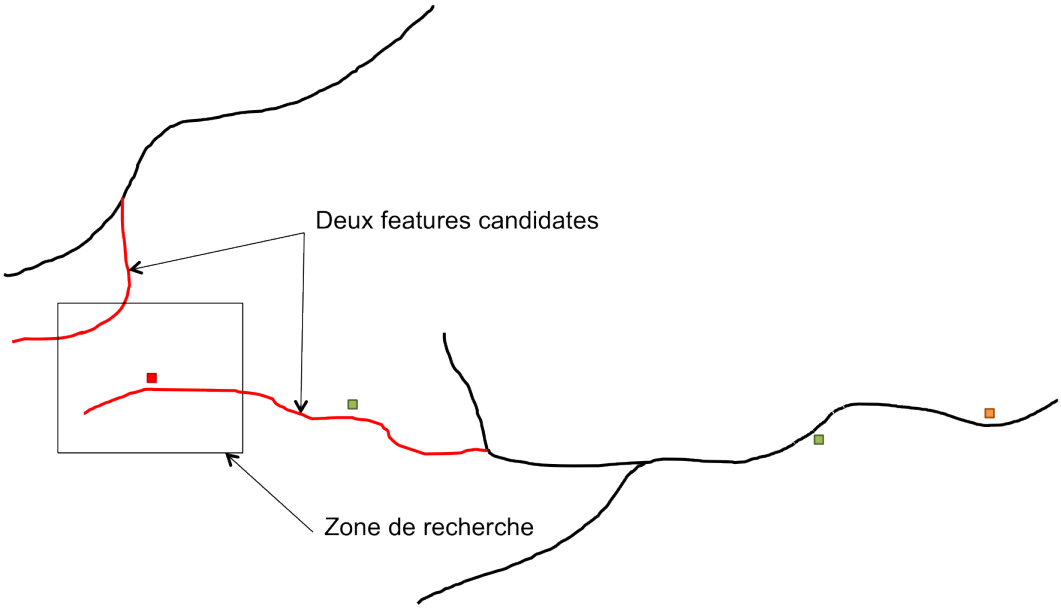

FIG. 9.5 : Schéma de la recherche du bief à l'extrémité du cours d'eau.

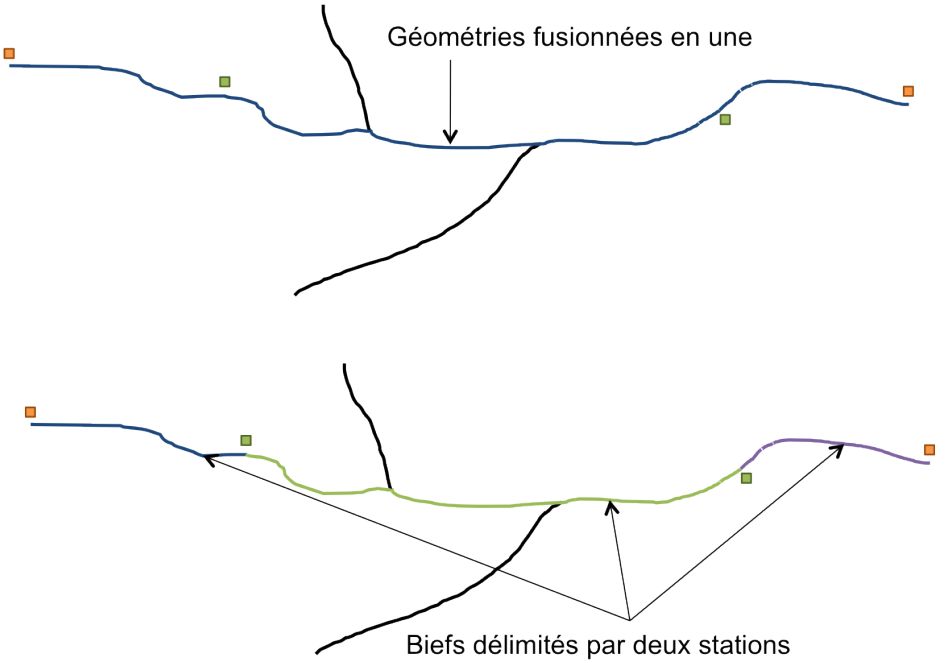

FIG. 9.6 : Schéma du découpage du cours d'eau en biefs délimités par des stations.

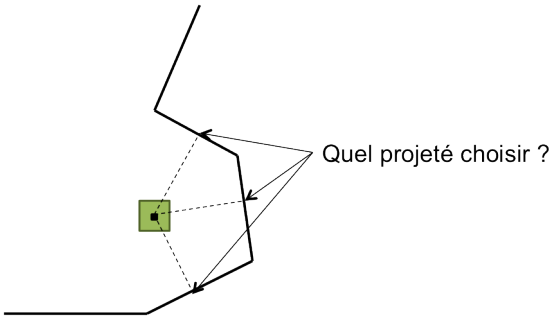

FIG. 9.7 : Schéma illustrant la difficulté du choix du point de projection d'une station.

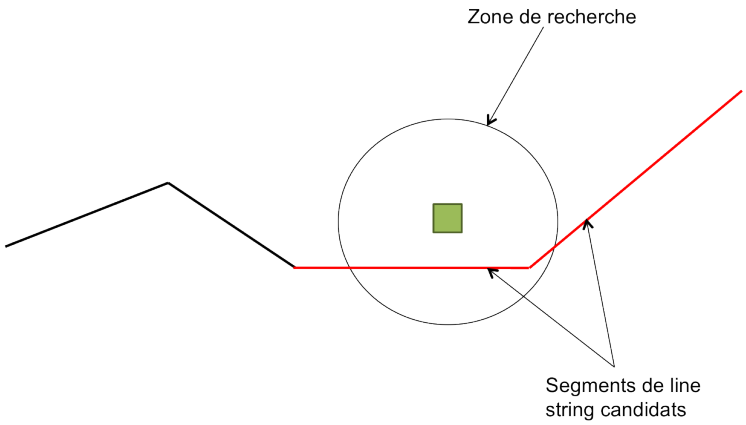

FIG. 9.8 : Schéma de la recherche des segments candidats d'une line string pour la projection orthogonale d'une station.

intersection avec une zone de recherche circulaire centrée sur la station (figure 9.8).

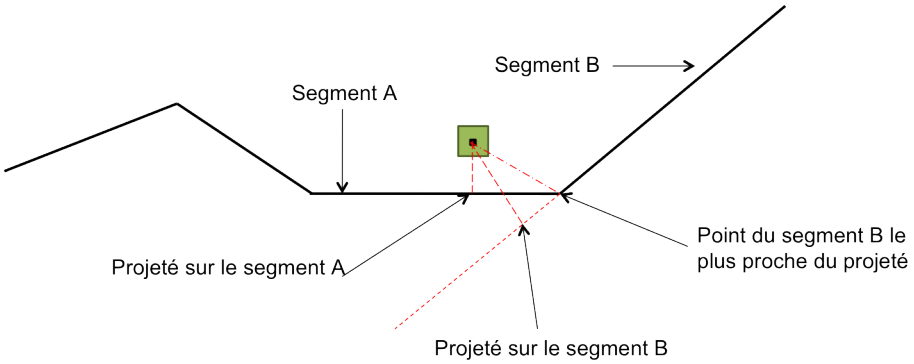

FIG. 9.9 : Schéma de projections orthogonales d'une station sur les segments candidats.

Le point projeté retenu est celui dont la distance avec la station est la plus courte, sur la figure 9.9 : le point projeté sur le segment A. S'il arrive que le projeté est en dehors du segment, l'extrémité du segment le plus proche du projeté le remplace.

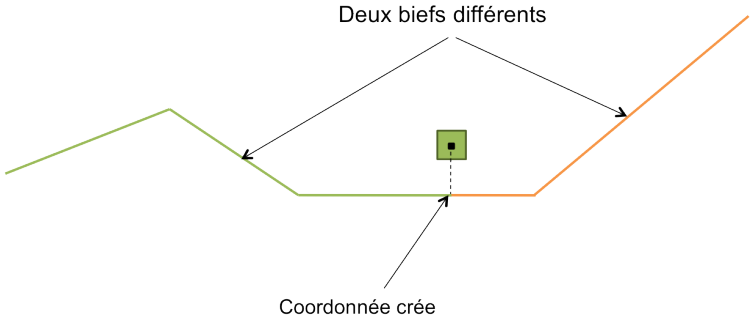

FIG. 9.10 : Schéma de découpage d'une line string au point de projection d'une station.

A l'endroit du point projeté, la line string est découpée en deux donnant naissance à deux nouveaux biefs délimités par deux stations (figure 9.10).
L'étape 10 voit la création, à l'aide de fabriques, des objets hydrologiques stations et reaches avec le calcul de quelques propriétés faites à partir des features extraites (ex : longueur des biefs). De même, c'est au cours de l'étape 10 qu'un objet project est créé pour rassembler les données hydrologiques.
Enfin, les étapes 11 et 12 sont chargées de créer les couches représentant graphiquement dans le SIG les stations et le cours d'eau (figure 9.11).

9.3 Extractions des données

Ces extractions ont pour but d'affecter des valeurs alphanumériques aux propriétés des objets hydrologiques (aspect alphanumérique des sites hydrologiques, voir chapitre 8 page 47). Ces valeurs alphanumériques sont issues de sources spatialisées comme des couches d'objets géographiques ou des images raster et de sources non spatialisées comme les bases de données de l'IRD.

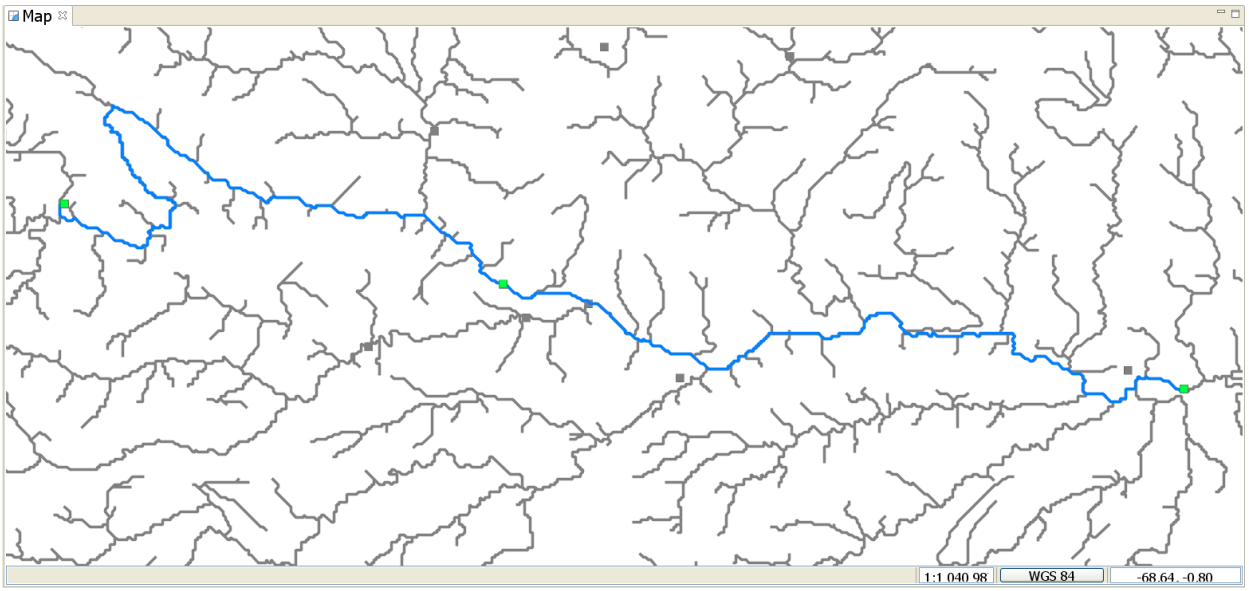

FIG. 9.11 : Capture d'écran d'Umodelis après extraction : en bleu le cours d'eau étudié, en vert ses stations, en gris les couches sources.

Les extractions sont paramétrables et mémorisables pour un traitement en lot : le module offre à l'utilisateur la possibilité de choisir la source de la donnée et la propriété de l'objet hydrologique à affecter, par exemple l'aire de drainage (drained area) des stations, et il lui permet de traiter automatiquement de la même manière tous les objets du même type à la manière d'une macrocommande.
La figure 9.12 montre une superposition d'une couche de stations sur une couche raster représentant les aires de drainages. Chaque pixel a pour attribut le nombre de pixels qu'il draine, c'est-à-dire le nombre de pixels en amont qui correspondent au ruissellement collecté par le pixel considéré. L'affectation de l'aire de drainage pour une station consiste à extraire l'aire de drainage du raster située sous les coordonnées de la station. Cette extraction utilise la forme géométrique de l'objet géographique. Le même principe est utilisé pour l'extraction de données alphanumériques provenant de couche d'objets géographiques, la donnée à extraire étant une propriété des features de la couche. Pour l'extraction depuis une base de données, le procédé s'appuie sur un formulaire d'interrogation de la base.

9.3.1 Gestion des métadonnées

Nous savons depuis le chapitre 2 que les métadonnées des sources de données ne sont pas normalisées et la masse d'informations et d'outils les exploitant sont tels qu'il n'est pas envisageable d'effectuer une normalisation. Cette normalisation doit donc être effectuée par Umodelis avec l'aide de l'utilisateur, le seul à connaître la source de données. Elle consiste à traduire les métadonnées de la source en métadonnées normalisées pour le système Umodelis.
La figure 9.13 montre un extrait de ces métadonnées d'Umodelis sous la forme d'énumérations pour chacune des classes d'objets hydrologiques. Bien que potentiellement fastidieux pour l'utilisateur, la traduction est heureusement mémorisée sous la forme de table de correspondance (mapping) métadonnées Umodelis/métadonnées source pendant la session de travail. D'autre part, cette figure est à associer à la figure 8.10

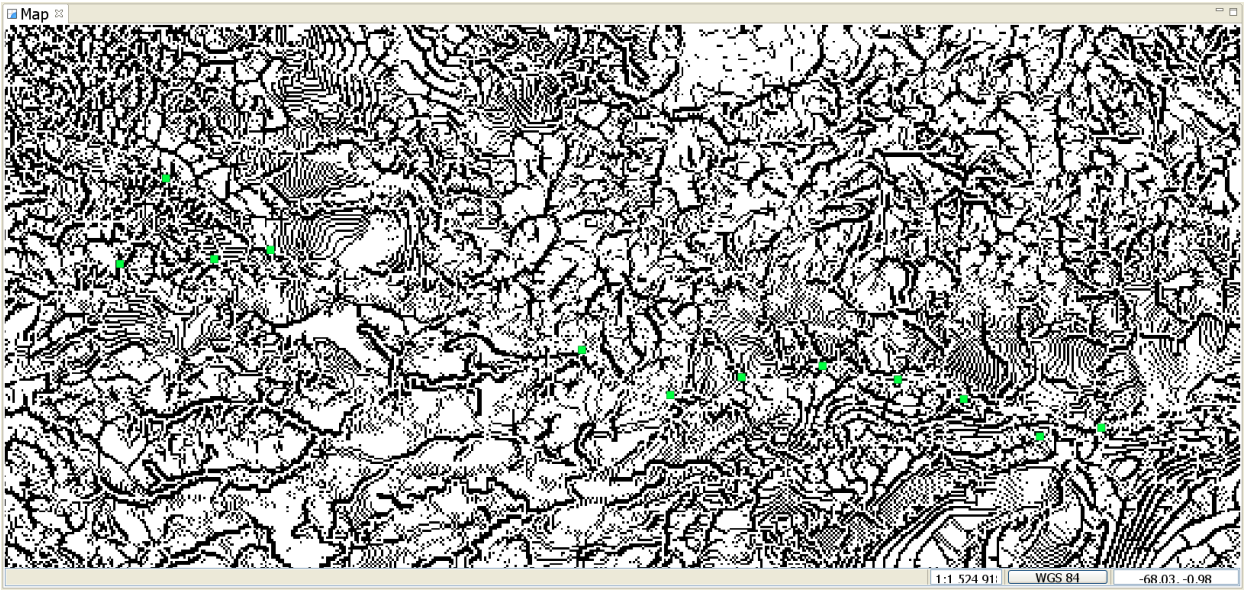

Fig. 9.12 : Capture d'écran de la superposition d'une couche raster (aires drainées) et d'une couche de stations hydrométriques (en vert sur la figure) dans Umodelis.

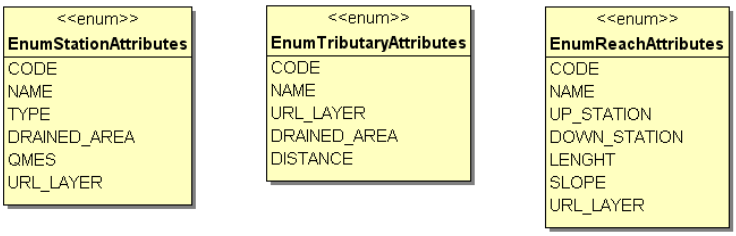

Fig. 9.13 : Extrait du diagramme de classes d'énumérations de propriétés d'objets hydrologiques.

page 59 : les métadonnées Umodelis sont les énumérations des propriétés des classes d'objets hydrologiques, elles leur servent d'identifiant. Les énumérations rendent possible l'association entre les propriétés des objets hydrologiques et les extractions mémorisées. Elles sont essentielles aux traitements génériques.

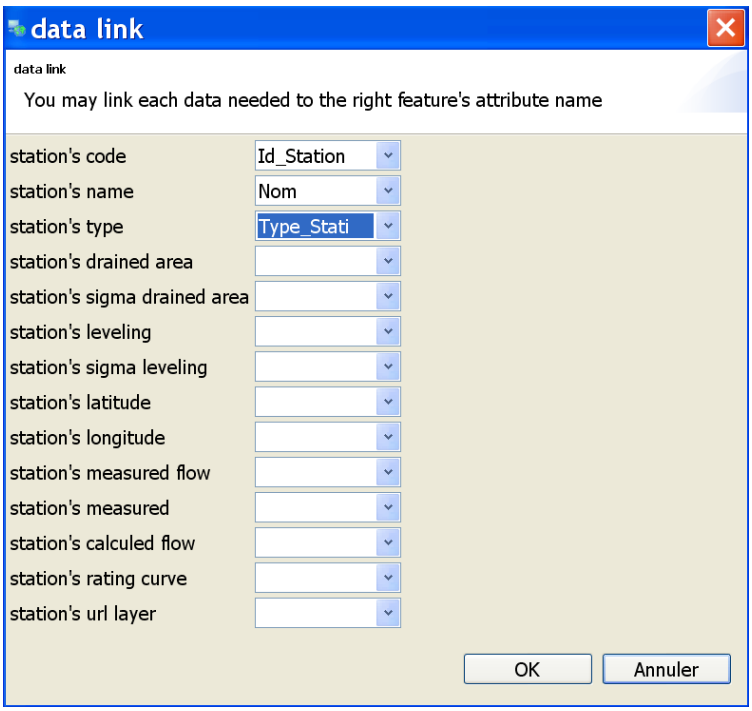

FIG. 9.14 : Capture écran de l'IHM de traduction de métadonnées de stations hydrométriques dans Umodelis.

La figure 9.14 présente un exemple d'interface de traduction de métadonnées de couche de stations en métadonnées Umodelis. Les utilisateurs doivent choisir pour chaque métadonnée Umodelis, une métadonnée source proposée.

La solution architecturale devra tenir compte de l'aspect configuration et généricité des extractions. Concernant le traitement générique des extractions, la solution retenue est une conception basée sur le patron commande (design pattern command ; GAMMA *et al.*, 1994 ; voir annexe A.3 page 106). La configuration est traduite par la mise en œuvre d'un assistant graphique de configuration (wizard), proposant aux utilisateurs les différentes sortes d'extractions ainsi que leurs paramètres afin de construire les extractions : les commandes issues du patron de conception commande.

9.3.2 Assistant de configuration des extractions

L'assistant de configuration d'extractions à pour but de construire, selon les choix de l'utilisateur, un objet ActionInt qui modélise la famille des extractions à effectuer. Le diagramme d'activité de la figure 9.15 montre le choix et la configuration des extractions. Ayant trois extractions possibles, l'utilisateur navigue dans l'une des trois zones d'activités (boîtes en pointillés) qui ont chacune pour finalité l'abandon de l'utilisateur ou la création d'un objet ActionInt contenant les paramètres de l'extraction à effectuer. Cet assistant

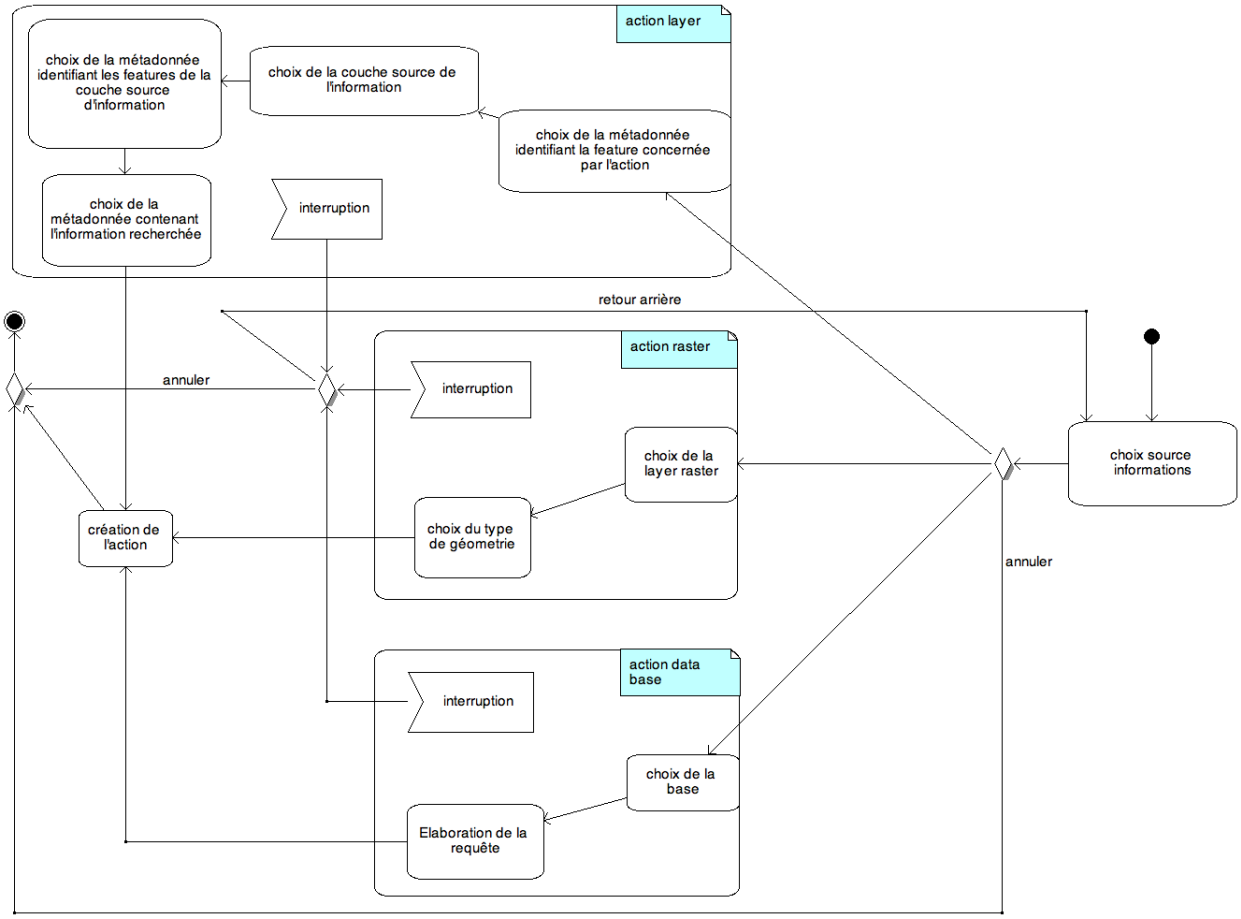

FIG. 9.15 : Diagramme d'activité de l'assistant de configuration (wizard) des extractions.

est basé sur le patron de conception fabrique (design pattern factory ; GAMMA *et al.*, 1994 ; voir annexe A.5 page 107) et est implémenté en utilisant l'assistant de configuration générique de la bibliothèque JFace d'Eclipse (portable en dehors d'Eclipse RCP).

9.3.3 Patron de conception commande et traitement par lot

Le patron de conception commande possède les qualités pour le traitement générique des extractions : son principe assure l'encapsulation des implémentations des extractions dans une méthode générique exécutable en lot par une classe cliente.

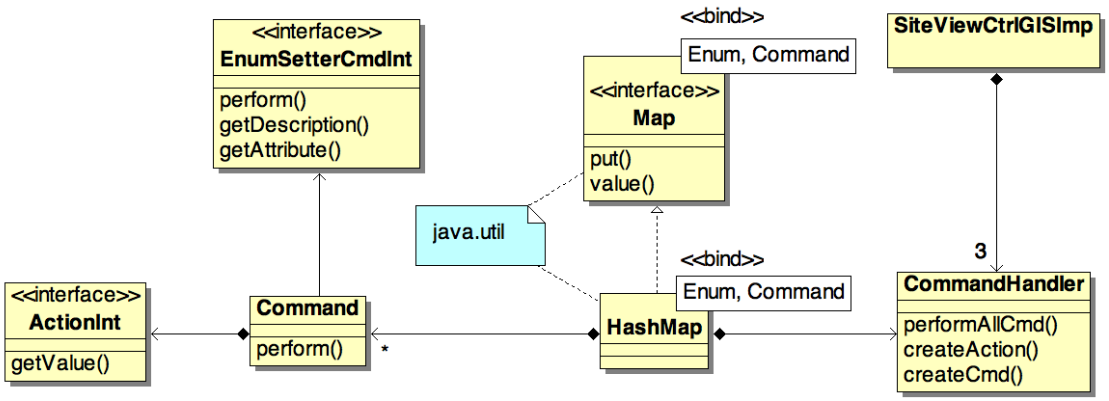

FIG. 9.16 : Extrait du diagramme de classes des commandes et des actions.

La figure 9.16 présente un extrait du diagramme de classes du patron de conception commande appliqué aux extractions. L'interface ActionInt représente l'abstraction des implémentations des extractions avec son unique méthode getValue renvoyant la valeur extraite. La classe Command est conçue pour traiter les opérations génériques d'extraction à l'aide d'un objet de type ActionInt et délègue à un objet de type EnumSetterCmdInt l'affectation de cette valeur à la propriété voulue d'un objet hydrologique donné. Les objets ActionInt et EnumSetterCmdInt étant injectés au moment de la construction de l'objet Command.
Les objets Command, un par propriété d'objet hydrologique sont appariés dans une table de correspondance (HashMap sur la figure) avec la métadonnée/énumération Umodelis à laquelle la propriété est associée. Le gestionnaire de commande CommandHandler, qui détient la table de correspondance, a pour rôle d'initier l'assistant de configuration par l'intermédiaire de sa méthode createAction, la construction de commandes par createCmd et l'exécution de toutes les commandes par performAllCmd pour l'exécution en lot. Le contrôleur SiteViewCtrlGISImp du module de création de site contient un gestionnaire pour chaque type d'objets hydrologiques.
L'implémentation du patron de conception commande diffère de sa définition classique (voir annexe A.3 page 106) : c'est une classe concrète Command qui encapsule l'interface des extractions au lieu de la réalisation d'une interface Command par des classes concrètes d'extractions. Cette variante permet d'avoir une interface d'extractions sans rapport avec le patron commande en prévision de la réutilisation des extractions en dehors du contexte de traitement par lot.
Les implémentations de l'interface EnumSetterCmdInt représentées sur l'extrait de diagramme de classes de la figure 9.17, montrent comment est associée une métadonnée Umodelis à une opération d'affectation sur une propriété d'un objet hydrologique.

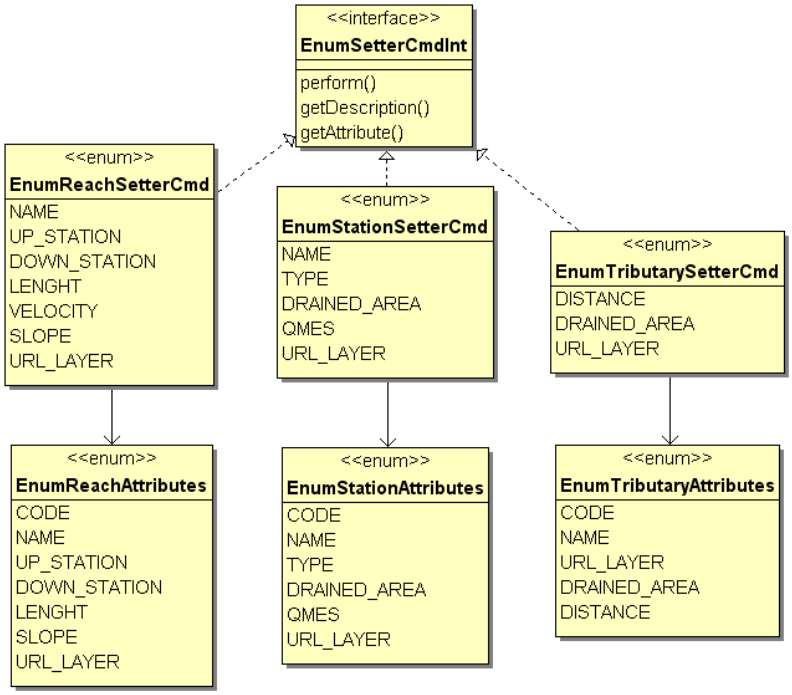

FIG. 9.17 : Extrait du diagramme de classes des énumérations d'opérations d'affectation de propriétés (setters) d'objets hydrologiques.

Chaque élément des énumérations d'opérations d'affectation de propriétés (setters) d'objets hydrologiques, réalise l'interface EnumSetterCmdInt et implémente la méthode perform. Cette méthode perform explicite l'affectation d'une valeur à une propriété d'un objet hydrologique à l'aide de ces méthodes commençant par set (voir la figure 8.10 page 59). La propriété de l'objet hydrologique à affecter, est déterminée par la métadonnée Umodelis utilisée pour définir l'élément de l'énumération.
Par exemple l'élément DRAINED_AREA de l'énumération EnumStationSetterCmd est défini à partir de l'élément de DRAINED_AREA de l'énumération des métadonnées des stations EnumStationAttributes et l'implémentation de sa méthode perform consiste à utiliser la méthode setDrainedArea de l'objet Stations donné. Ainsi les méthodes d'affectation des propriétés des objets hydrologiques sont abstraites en éléments d'énumérations.
La figure 9.18 présente les trois extractions de valeurs alphanumériques proposées. ActionGridImp et ActionCentroidPointGridImp implémentent respectivement les extractions raster (grid) basées respectivement sur tous les points de la géométrie d'un objet géographique ou sur son centroïde. ActionFeatureImp effectue l'extraction de la valeur d'une propriété d'une feature située sur une couche donnée et ActionDBImp effectue une extraction en base de données. Les extractions n'étant pas spécialisées par types d'objets géographiques qu'elles modifient, elles délèguent l'accession aux propriétés des objets géographiques (features) à un objet de type EnumGetterCmdInt fourni au moment de leur construction.
EnumGetterCmdInt dont les implémentations sont données à la figure 9.19, s'appuie sur le même mécanisme que l'interface EnumSetterCmdInt pour abstraire les méthodes d'accession aux propriétés des objets hydrologiques en éléments d'énumérations.
La figure 9.20 illustre la dynamique de la création d'une commande d'extraction de données raster pour

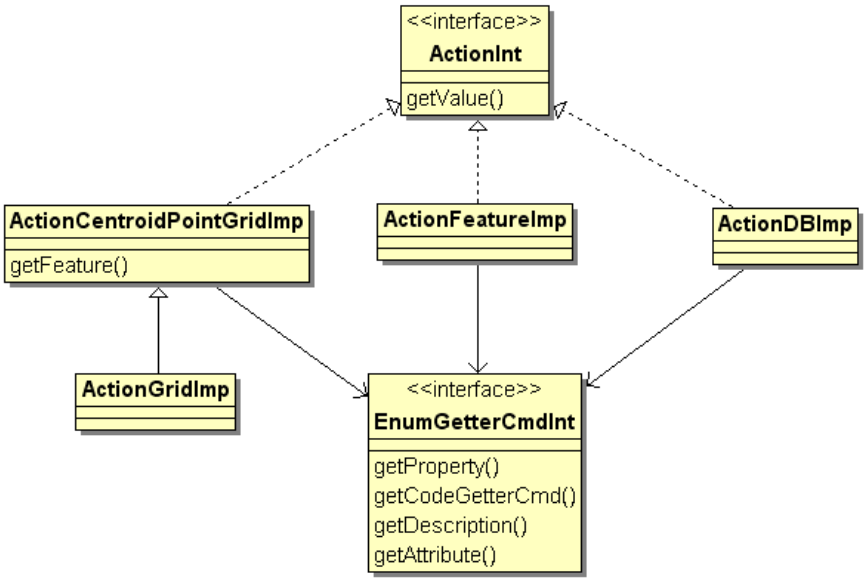

FIG. 9.18 : Extrait du diagramme de classes des actions.

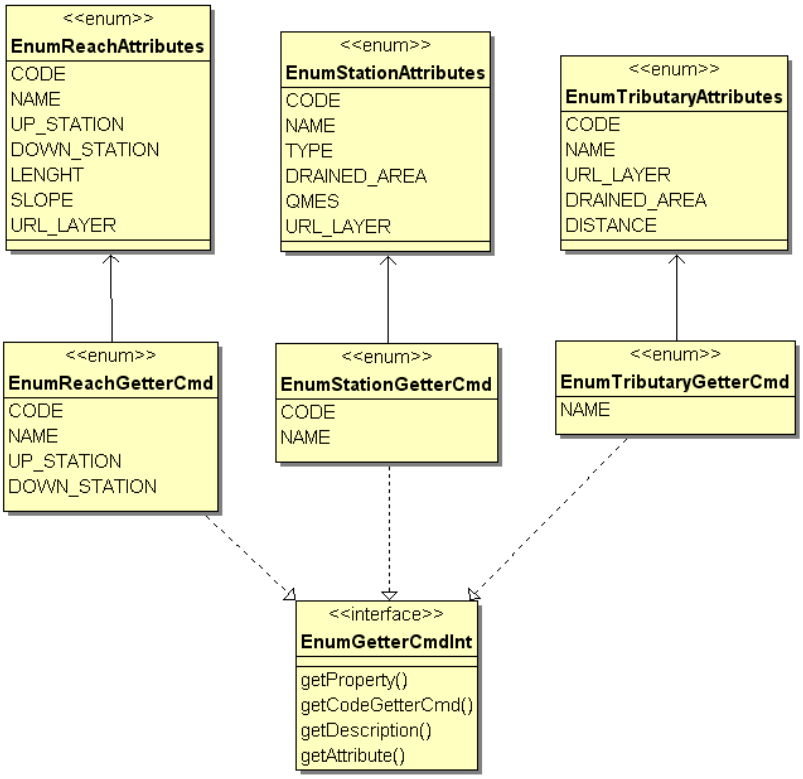

FIG. 9.19 : Extrait du diagramme de classes des énumérations d'opérations de lecture de propriétés (getters) d'objets hydrologiques.

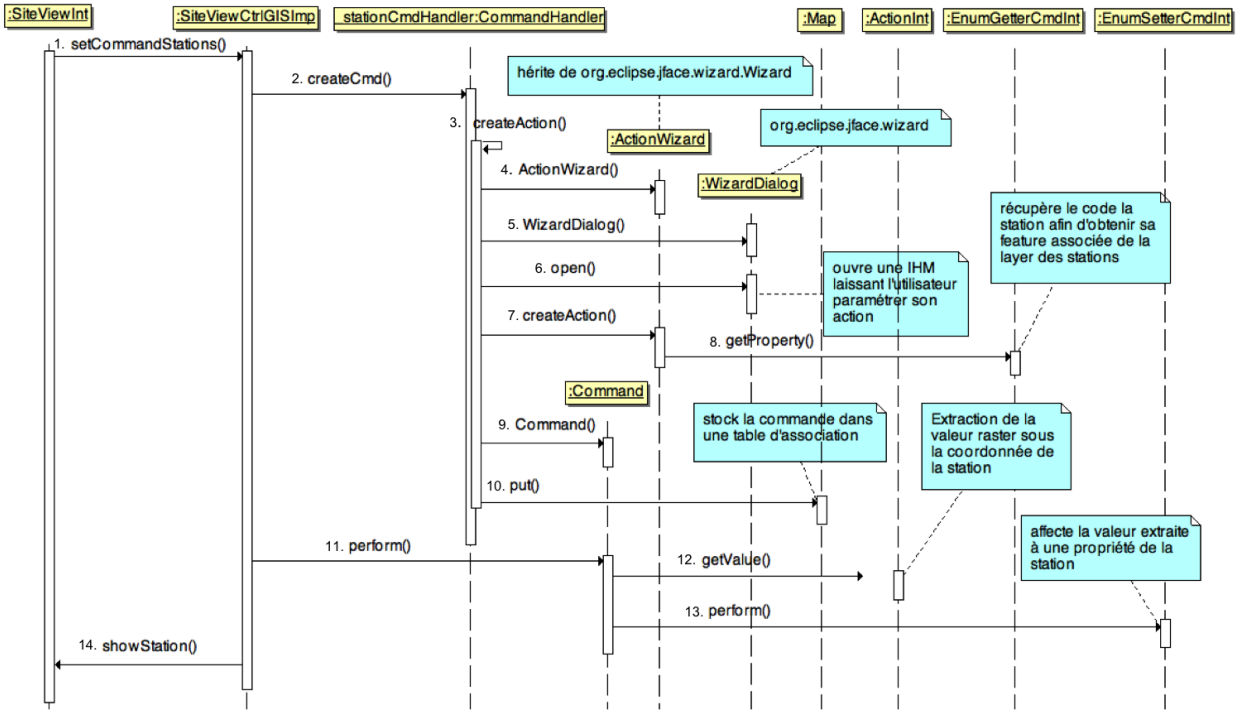

FIG. 9.20 : Extrait du diagramme de séquences de la création d'une commande de stations.

une station puis de son exécution. Elle se décompose en trois parties : la création d'un objet ActionInt avec les séquences 1 à 7 qui comportent l'exécution de l'assistant de configuration (5 et 6), la création de la commande (9) et l'exécution de la commande créée avec les séquences 11 à 13.
Ce diagramme de séquences, à la figure 9.21, montre l'exécution en lot des commandes mémorisées pour une station hydrométrique par itération (étape 4) des commandes mémorisées dans la table de correspondance (Map).
Enfin, la capture d'écran de la figure 9.22 illustre l'intégration du mécanisme de commande d'extraction. Les boutons [...] incorporés dans l'IHM d'édition des objets hydrologiques permettent d'ouvrir l'assistant de configuration pour la création et la mémorisation (ou le remplacement) d'une extraction de données à affecter à la propriété près de laquelle ils sont situés.

9.4 Gestion des sites hydrologiques

La gestion des sites regroupe les opérations d'ajout, de suppression et d'édition de sites hydrologiques ainsi que la gestion des projets (cas d'utilisation figure 9.2). Cette gestion fait appel à un ensemble de gestionnaires accessibles aux contrôleurs du module. La gestion est organisée en deux parties : les gestionnaires d'objets hydrologiques puis ceux de projets.

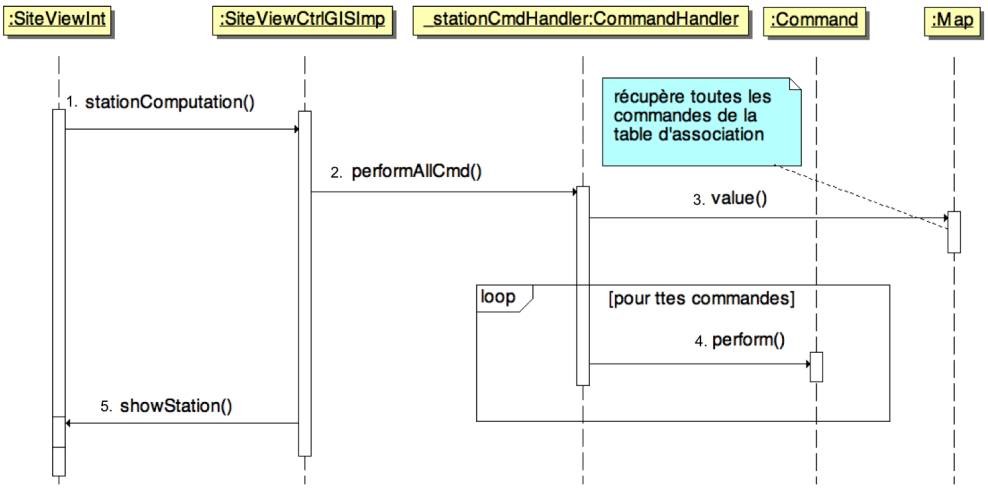

FIG. 9.21 : Extrait du diagramme de séquences du traitement par lot de commandes.

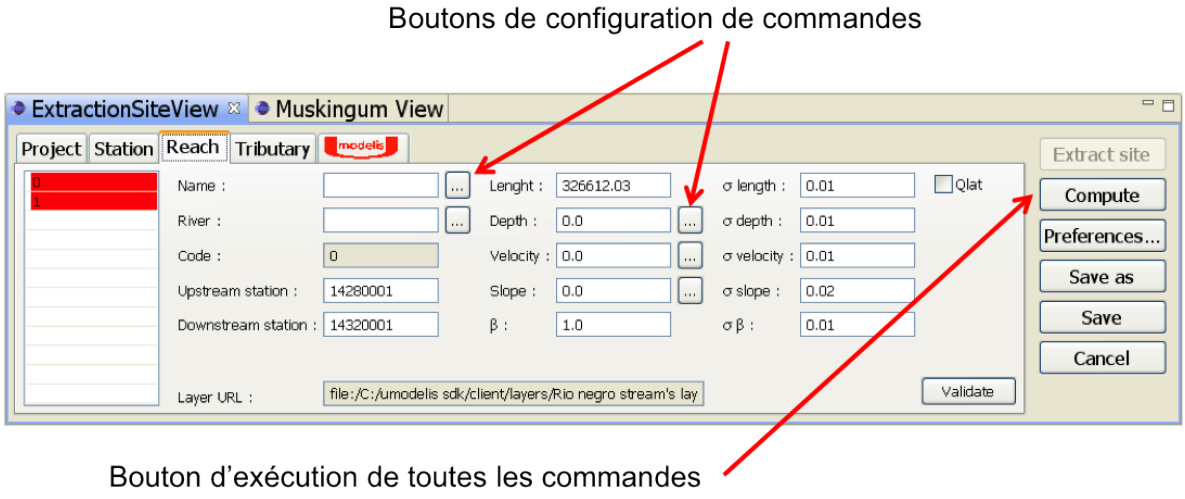

FIG. 9.22 : Capture d'écran de l'IHM d'édition des biefs dans Umodelis.

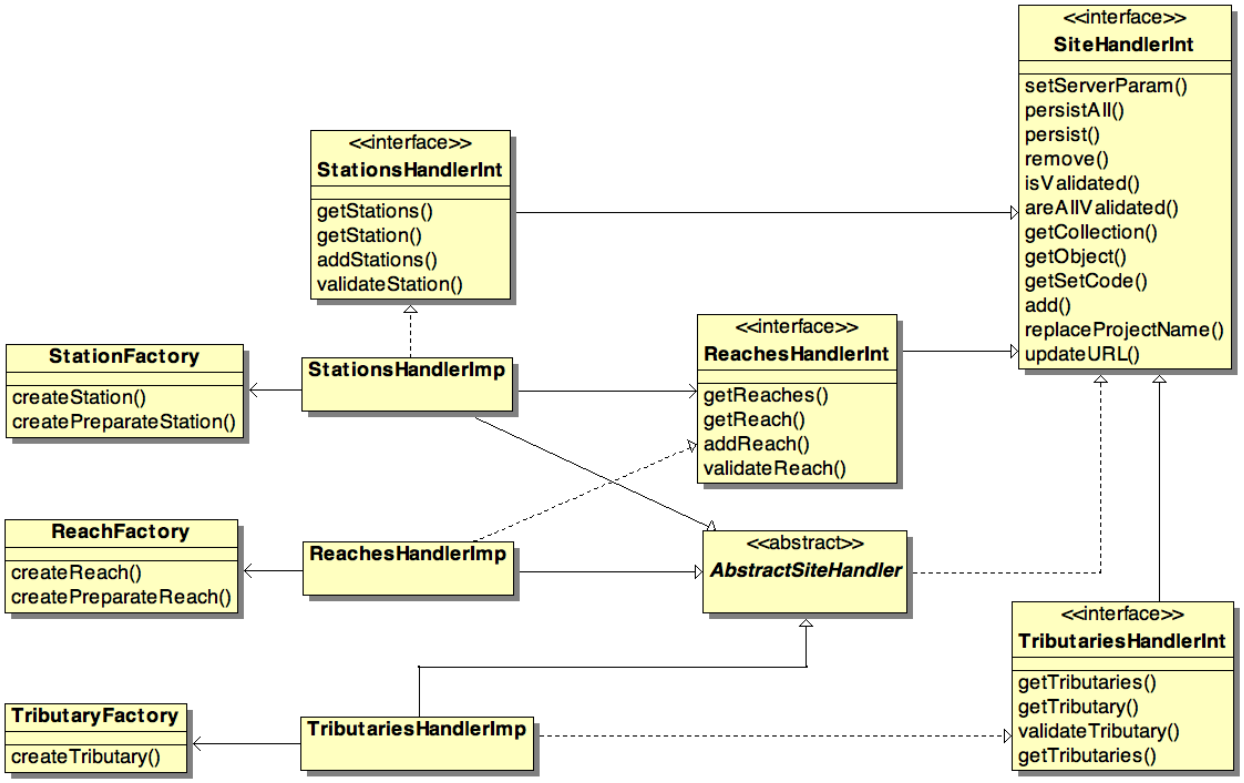

FIG. 9.23 : Extrait du diagramme de classes des gestionnaires d'objets hydrologiques.

9.4.1 Gestionnaires d'objets hydrologiques

La figure 9.23 présente le diagramme de classes des gestionnaires d'objets hydrologiques. L'architecture des gestionnaires est basée sur leur spécification par l'interface SiteHandlerInt. Cette interface simplifie l'insertion de gestionnaires de nouveaux types d'objets hydrologiques (par exemple le bassin versant) et offre la possibilité de traitements génériques pour le contrôleur du module.
La classe abstraite AbstractSiteHandler centralise les méthodes communes des gestionnaires afin de faciliter la maintenance. Les interfaces des gestionnaires d'objets hydrologiques (StationsHandlerInt, etc.) et leurs implémentations gèrent la spécificité de chaque type d'objets hydrologiques en encapsulant les règles de persistance ou les règles de vérifications des données.
Chaque type d'objets hydrologiques est accompagné d'une fabrique afin de déléguer leur construction et de rendre les gestionnaires robustes face à d'éventuelles modifications de leur structure (ceci est également valable pour les extractions clientes de ces fabriques).

9.4.2 Gestionnaires de projets

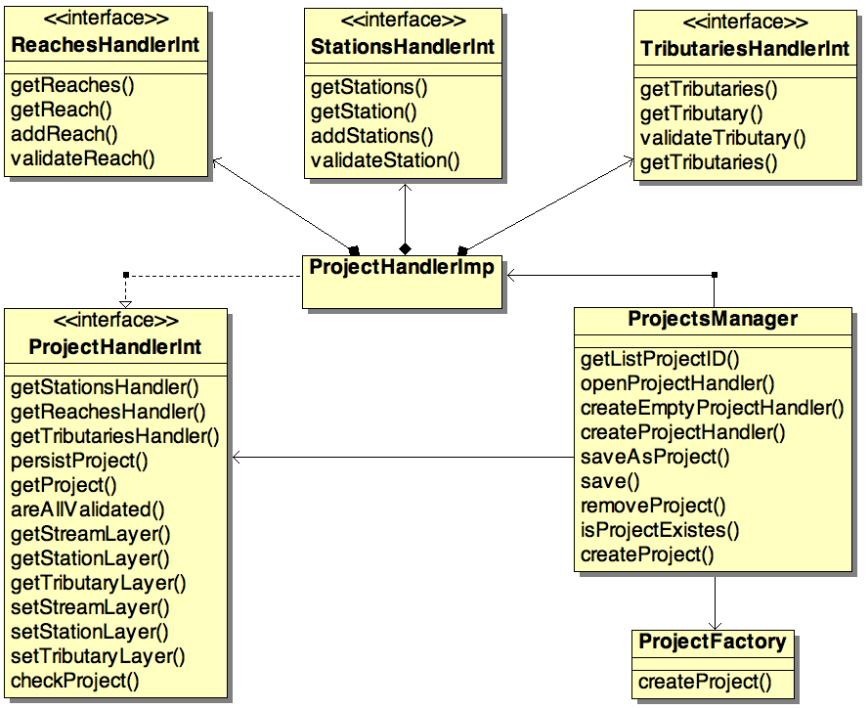

FIG. 9.24 : Extrait du diagramme de classes du système de gestion de projets.

Sur la figure 9.24, l'interface ProjectHandlerInt et son implémentation ProjectHandlerImp représentent le gestionnaire de projets qui assure la cohérence entre un projet et les gestionnaires des sites hydrologiques. Il contient la logique de sauvegarde en cascade des sites et du projet associé, la logique de vérification du projet et il maintient un lien vers les couches de stations, biefs et tributaires si le client est intégré à un SIG. Concernant la vérification du projet, le patron de conception état (design pattern state ; GAMMA *et al.*,

1994 ; voir annexe A.4 page 106) n'a pas été jugé pertinent lors de la conception, un projet n'ayant que deux états possibles : valide et invalide.

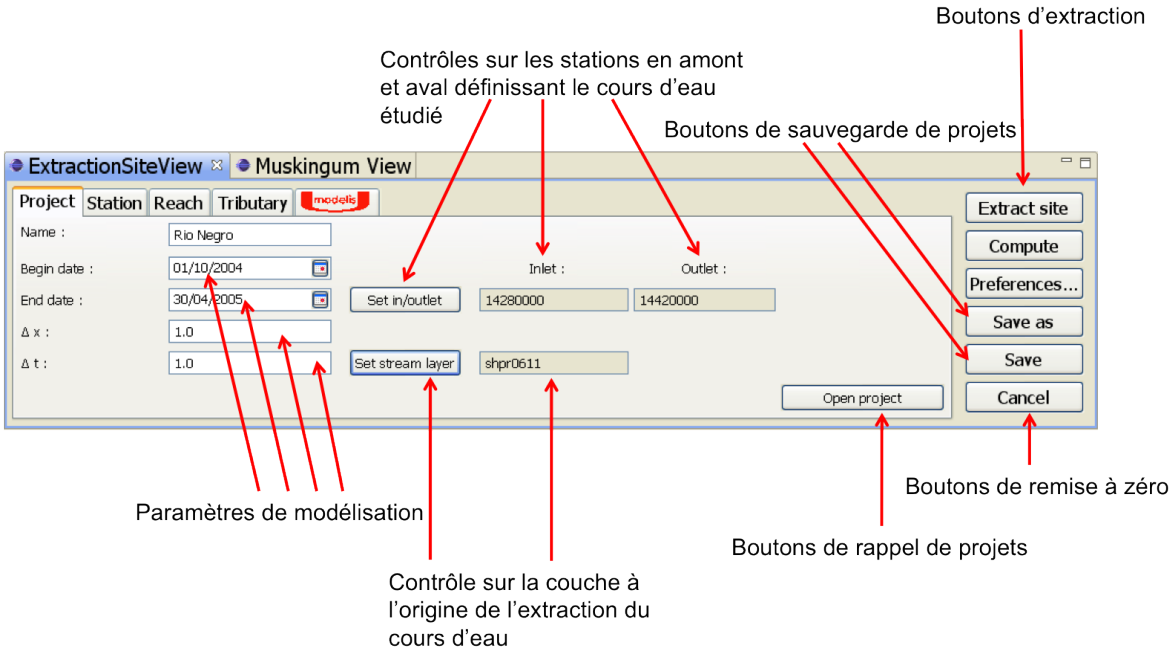

FIG. 9.25 : Capture d'écran de l'IHM d'édition des projets dans Umodelis.

ProjectManager est la classe responsable de la connexion avec le serveur de modélisation : elle joue le rôle de contrôleur lors de la sauvegarde, du rappel ou de l'effacement d'un projet et est responsable de la création d'objets Projects en s'appuyant sur la fabrique ProjectFactory.
Enfin, Umodelis expose une IHM qui propose les services des gestionnaires à l'utilisateur. La figure 9.26 montre l'IHM d'édition des stations et la figure 9.25 celle des projets.

9.5 Conclusion

Le module de création de sites hydrologiques apporte l'aide à la préparation des données en entrée de modèles hydrologiques linéaires. Il établit également les bases de l'extraction des sites hydrologiques provenant des couches d'objets géographiques SIG, le traitement par lot des extractions de données grâce à la gestion des métadonnées des sources de données et l'abstraction des accès aux propriétés des objets hydrologiques ainsi que la gestion et l'édition des objets hydrologiques et des projets. Le prochain chapitre portera sur le module d'exploitation de résultats chargé de contrôler les simulations des modèles hydrologiques et d'exploiter leurs résultats.

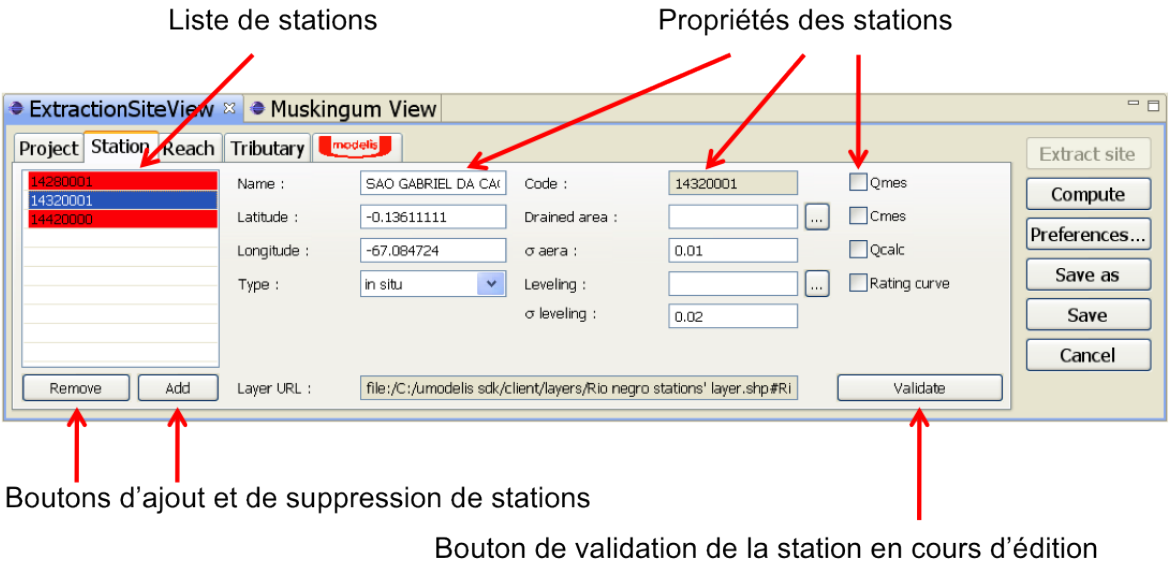

FIG. 9.26 : Capture d'écran de l'IHM d'édition des stations hydrométriques dans Umodelis.

CHAPITRE 10

Module d'exploitation de résultats

Le module d'exploitation est le deuxième module développé pour la plateforme Umodelis. Il offre à l'utilisateur à la fois une interface de contrôle de simulations pour les modèles hydrologiques linéaires et une interface de dépouillement de données en sortie de modèle. Ce chapitre présente les cas d'utilisations du module puis montre l'intégration d'un outil existant au sein de la plateforme.

10.1 Cas d'utilisations

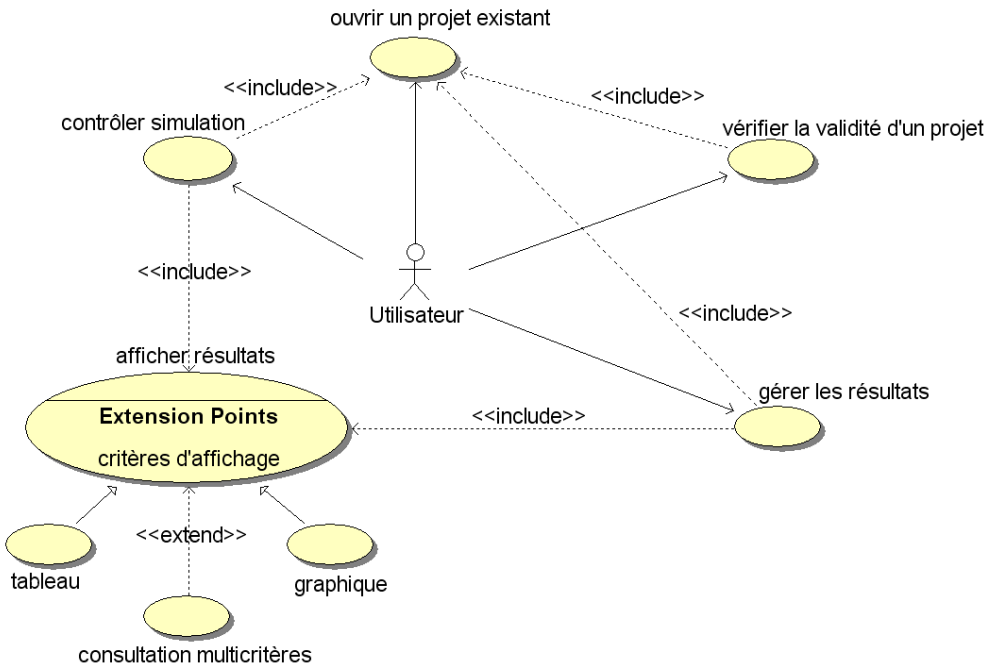

FIG. 10.1 : Diagramme de cas d'utilisation du module d'exploitation de résultats.

Comme le montre la figure 10.1 le module d'exploitation de résultats propose à l'utilisateur de choisir un projet de site hydrologique (cas ouvrir un projet existant) et d'y appliquer un algorithme de modélisation (cas contrôler simulation) : démarrer une simulation, l'annuler ou afficher la progression de l'algorithme. Le cas contrôler simulation sera traité au cours du prochain chapitre dans le cadre de la communication entre les clients et le serveur de modélisation.

L'affichage des résultats est modélisé par le cas d'utilisation afficher résultats dont les présentations sous forme de tableau ou de graphique sont des spécialisations. Le cas consultation multicritères étend le cas afficher résultats et propose à l'utilisateur de trier l'information selon des critères (capteur, type de série chronologique, résolution temporelle, etc.) proposés par Umodelis.
Le cas vérifier la validité d'un projet est responsable de la validation du projet par rapport aux règles du modèle hydrologiques choisi. Cette validation est complémentaire avec celle du module de création de sites hydrologiques qui vérifie la cohérence du projet et des sites.
Enfin, le cas gérer les résultats regroupe les opérations de sauvegarde et de rappel des résultats obtenus. Il n'est actuellement pas implémenté mais les résultats étant des objets hydrologiques, sa réalisation réutilisera les gestionnaires d'objets du module de création de sites (voir section 9.4.1 page 77).

10.2 Intégration d'un outil existant

Préalablement, les chercheurs et ingénieurs de l'IRD Brésil (Marie-Paule Bonnet, Mathilde Cauhopé, Gérard Cochonneau) ont développé un prototype de logiciel de simulation hydrologique Muskingum-Cunge. Il est écrit en Java et utilise des structures de données proches de celles d'Umodelis. Néanmoins, il n'offre aucune des facilités d'Umodelis : pas de distribution des modèles et des données, ni de lien avec les bases de données (les données sont décrites dans des fichiers textes locaux), pas d'interface d'édition des données (l'utilisateur doit éditer lui même les fichiers) et aucune représentation graphique des concepts géomatiques (pas de SIG).
En outre, ce prototype a été conçu uniquement pour le modèle Muskingum-Cunge. Toutefois, il a servi de base de travail pour le développement du module d'exploitation des résultats et pour l'implémentation du modèle Muskingum-Cunge sur le serveur de modélisation (voir chapitre 11 page 83). Les points suivants traitent du travail d'intégration de cet outil dans la plateforme par l'équipe de développement.

10.2.1 Réutilisation des composants du prototype

Parmi les composants du prototype, l'agencement de l'IHM a été réutilisé pour son implémentation en SWT. Le contrôleur (MuskingumManager) ainsi que la logique applicative (les classes SerieTemp) ont également été intégrés (voir figure 10.2). Le point suivant traite des modifications apportées au prototype pour son intégration dans la plateforme.

10.2.2 Modifications apportées

Comme indiqué dans l'introduction de la section, le prototype est une application locale accédant à des données uniquement sur les systèmes de fichiers. La méthode d'intégration du prototype est relativement simple. Dans un premier temps, l'équipe de développement d'Umodelis a isolé le code relatif aux traitements des données pour l'adapter à la bibliothèque Umodelis. Cette opération a été facilitée grâce aux sémantiques proches des deux structures de données. Puis, l'équipe a procédé à l'implémentation de l'interface de contrôle de simulation (abordé au chapitre 11 page 83), la gestion de la connexion avec le serveur de modélisation et la gestion de sauvegarde des résultats.
Enfin, comme pour le module de création de sites hydrologiques (voir section 7.2 page 45), un point d'extension d'uDig a été créé, représenté sur la figure 10.2 par la classe MuskingumView, héritant de ViewPart,

point d'entrée abstrait d'Eclipse RCP. L'IHM est intégrée par héritage de la classe Composite faisant partie de la bibliothèque SWT.

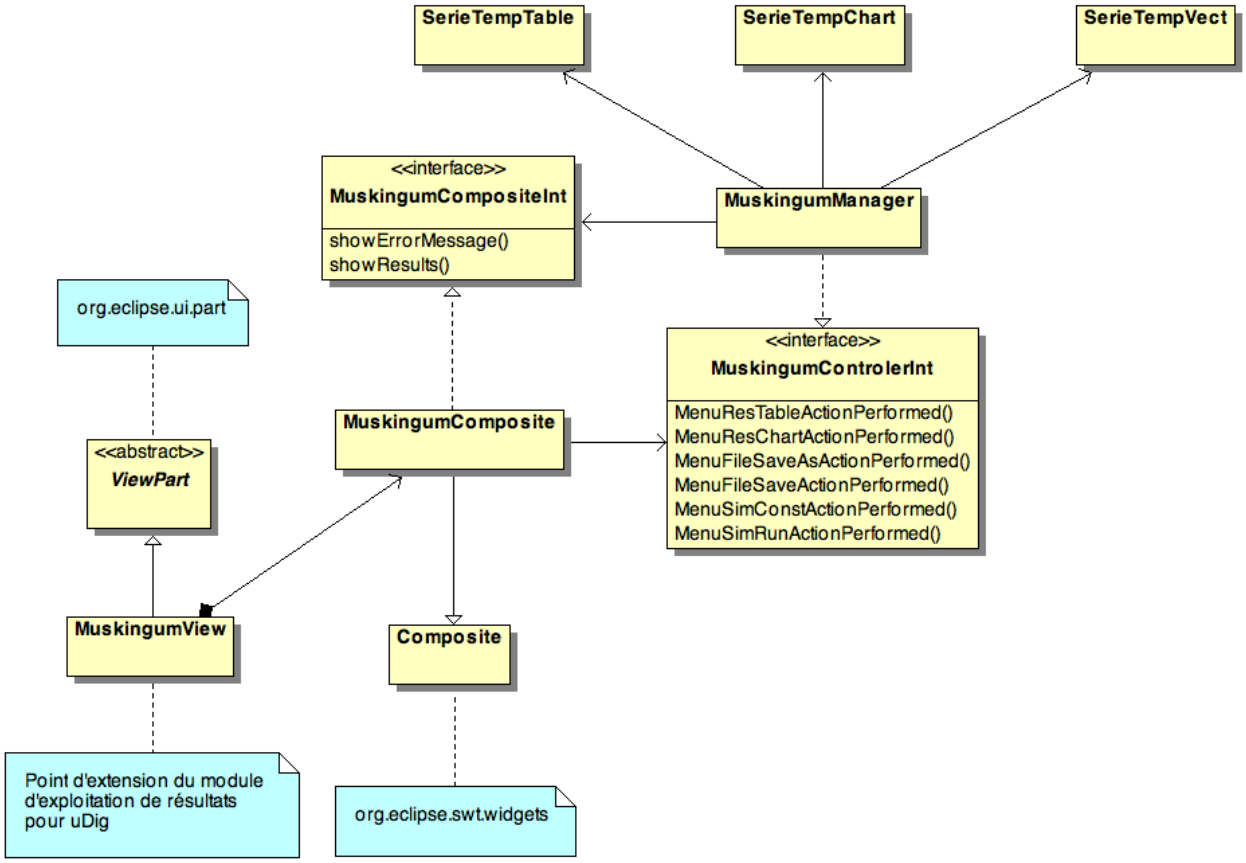

FIG. 10.2 : Extrait du diagramme de classes modélisant l'intégration du module d'exploitation de résultats dans uDig.

Comme l'indiquent les inclusions multiples de cas d'utilisations du module (caractère séquentiel du module), l'équipe a envisagé une conception plus robuste basée sur l'utilisation du patron de conception état (design pattern state ; GAMMA *et al.*, 1994 ; voir annexe A.4 page 106). Malheureusement, par manque de temps, elle n'a pas été menée mais fait partie des futurs améliorations de la plateforme.

10.3 Conclusion

Une simple adaptation du code a permis de réutiliser en grande partie un outil existant. La plateforme montre un exemple de réutilisation rendue possible grâce à sa conception modulaire. Le prochain chapitre a pour sujet l'architecture du serveur de modélisation ainsi que les interactions avec les modules clients d'Umodelis.

Chapitre 11

Serveur de modélisation hydrologique

Dernière partie de la plateforme Umodelis, le serveur de modélisation hydrologique cumule les rôles d'accès et de persistance des données hydrologiques et de simulations hydrologiques. Le chapitre commence par un rappel sur les rôles et les choix de conception du serveur pour continuer sur la décision d'utiliser l'architecture Java 2 Enterprise Edition (J2EE), sa description et son déploiement. Il détaille également la conception du module d'accès et de persistance des données hydrologiques, l'intégration des modèles hydrologiques et le contrôle de la simulation et propose une discussion sur la gestion des ressources de calcul.

11.1 Rôles du serveur de modélisation

Vus à la section 5.3.2 page 31, les composants du serveur de modélisation hydrologiques sont la persistance et l'accès aux données hydrologiques ainsi que les modèles hydrologiques.

11.1.1 Persistance et accès aux données

Le serveur de modélisation offre un mécanisme de distribution des structures de données de la plateforme Umodelis afin d'en garantir l'accès depuis n'importe quel poste. Ce mécanisme consiste en l'extraction des données hydrologiques des bases de données de l'IRD puis leur encapsulation dans les objets hydrologiques (voir figure 8.10 page 59) et l'opération inverse : la désencapsulation des données puis leur persistance en base de données. Enfin, le module met à disposition un contrôleur qui permet aux modules client d'Umodelis de consulter les données.

11.1.2 Modèles hydrologiques

La famille de composants modèles hydrologiques concerne l'intégration des algorithmes de modélisation hydrologique dans la plateforme Umodelis et le contrôle des simulations des modèles par les clients d'Umodelis. La conception de ce type de composant aboutit à la spécification d'un patron de modèles.

11.1.3 Spécifications architecturales

Comme évoqué au chapitre 5 page 27, le serveur de modélisation doit promouvoir l'aspect distribué et orienté composant d'Umodelis. Une attention particulière est portée aux services rendus par l'intergiciel (middleware) sous-jacent au serveur, notamment la sécurité d'accès au serveur et aux applications, la sûreté des transactions et l'abstraction de la persistance des données pour les modules des clients. Ces points éclairent le choix de l'architecture du serveur, abordé dans la section suivante.

11.2 Choix de l'architecture J2EE

Parmi les intergiciels opérationnels du marché, l'équipe de développement a évalué seulement ceux orientés objets afin d'éviter une couche d'emballage (wrapper) supplémentaire entre les objets des clients et les structures de données du serveur. La plateforme J2EE a retenu notre attention pour sa robustesse, J2EE est largement déployée dans les systèmes d'information d'entreprise et compte quelques années d'exploitation, pour sa pérennité, J2EE est le fer de lance de Sun Microsystems et est soutenue par plusieurs acteurs majeurs de l'informatique dont IBM, et enfin pour sa documentation et ses tutoriels très faciles d'accès. D'autre part l'équipe de développement en possédait déjà quelques connaissances. J2EE répond aux spécifications d'Umodelis en termes de services et argument décisif, elle est écrite en Java, le même langage d'implémentation que les clients Umodelis.
Les autres alternatives comme les implémentations libres de droits de Common Object Request Broker Architecture (CORBA) : Tao, JacORB, OpenORB, etc. n'ont pas été retenues à cause de la relative plus forte complexité de mise en œuvre par rapport à J2EE. La solution de Microsoft .NET a été écartée à cause de la contrainte de gratuité des environnements de développement et de non portabilité officielle voir le chapitre 2 page 9 (même si le projet Mono assure une portabilité Mac OS X, Linux et Windows mais non supportée par Microsoft).

11.2.1 Services J2EE

Pour résumer, J2EE couvre les services suivants (SUN-MICROSYSTEMS, 2008) :

- Sécurité :

J2EE implémente les principaux paradigmes de sécurité comme l'authentification des utilisateurs, la gestion des autorisations, l'intégrité et la confidentialité des données. J2EE gère les politiques de qualité de services (QoS) et protège du déni de service (DoS). La sécurité s'applique également aux applications Web et Java hébergées au sein de J2EE et aux systèmes d'information d'entreprise (EIS), les bases de données concernant Umodelis.

- Gestion d'événements ou système de messagerie :

J2EE propose un système de messagerie ou Message-Oriented Middleware (MOM) avec son API Java Message Service (JMS) qui est le moteur de la gestion des événements de J2EE : les Message-Driven Beans (MDB).

- Transaction :

Transaction est un service garantissant l'intégrité du traitement des requêtes faites au système. Un système de rollback renforce le déterminisme post-transaction.

- Connexion aux EIS :

Service chargé d'assurer la connexion aux l'EIS par le biais de Java Data Base Connectivity (JDBC) pour le cas des bases de données et l'injection de dépendances dans les objets représentant les données en bases par un jeu d'annotations (javax.annotation).

- Gestion de cycle de vie :

Elle s'occupe de l'accès concurrentiel des clients à la couche applicative et la couche données du serveur en gérant l'allocation dynamique des objets distribués stockés dans une réserve (pool).

- Nommage et invocation :

Les objets distribués sont accessibles au moyen du service de nommage Java Naming and Directory Interface (JNDI) et l'invocation de leurs méthodes est basée sur le protocole Remote Method Invocation - Internet Inter-Orb Protocol (RMI-IIOP), compatible avec celui de CORBA.

11.3 Architecture J2EE du serveur

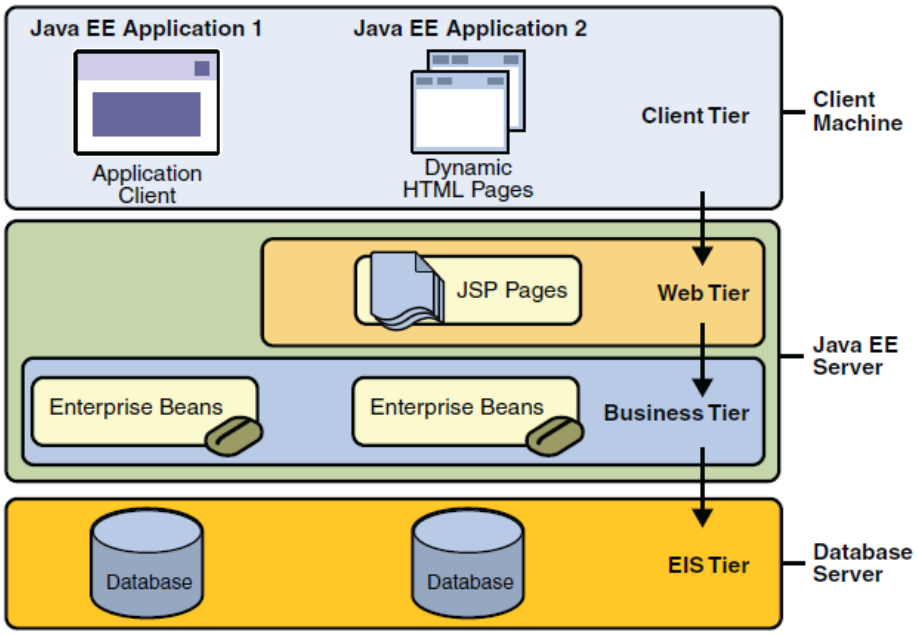

FIG. 11.1 : Organisation en couches de l'architecture J2EE (d'après SUN-MICROSYSTEMS, 2008).

J2EE est une architecture multi tiers où l'on distingue trois couches illustrées à la figure 11.1. Les applications Java et les clients légers Web forment la couche client qui se connecte à la couche J2EE serveur composée du tiers Web responsable du traitement des requêtes des clients Web et du tiers logique applicative (business tiers sur la figure). Enfin, la couche J2EE est connectée à la couche donnée de l'entreprise dont elle assure l'abstraction auprès de la couche client.
La figure 11.2 illustre les flux de communications entre les composants des couches et l'organisation selon le patron MVC modèle 2 de J2EE. Suivant le type de client, la topologie MVC n'est pas la même : pour les clients légers, la génération de la vue (pages JSP) et leurs contrôleurs (Servlets) sont situés dans le container Web, le navigateur Web du client est uniquement chargé du rendu graphique des pages et de la connexion avec le container Web. Concernant les clients lourds (application cliente), la vue et le contrôleur sont situés au niveau du client.
Pour les deux types de client, le modèle est situé dans le container d'Enterprise JavaBeans (EJB) : il est constitué de différentes entités porteuses de logiques applicatives et représentant les données d'entreprise. Pour le client lourd, le container d'EJB est contacté directement (en cas d'absence de pare-feu) tandis que pour le client léger (et le client lourd en cas de présence de pare-feu), il est atteignable par l'intermédiaire

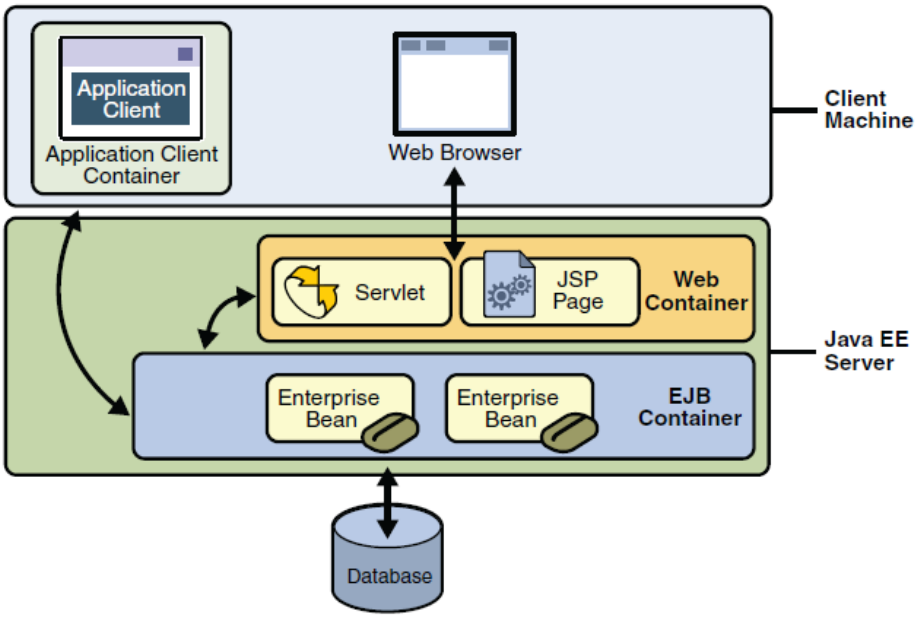

FIG. 11.2 : Communication au sein de J2EE (d'après SUN-MICROSYSTEMS, 2008).

des Servlets. Enfin, le container d'EJB est responsable de la représentation dynamique des données en bases.
La figure 11.3 schématise les entités du container EJB. Il existe trois sortes d'EJB :

- Entity Beans :

Ils représentent les données au sein d'un mécanisme de persistance. Ils modélisent les données en base : la classe d'un entity bean représente une table, ses instances les enregistrements de la table. Le serveur J2EE maintient une correspondance dynamique entre les deux représentations afin que l'une soit l'exact reflet de l'autre.

- Session Beans :

Ils encapsulent la logique applicative (business logic) et accèdent aux entity beans. Selon leur nature, ils gardent l'état de la conversation avec le client (stateful) ou au contraire ils ne retiennent aucune mémoire des échanges avec le client (stateless) et sont interchangeables entre les clients.

- Message-Driven Beans :

Basés sur le système de messagerie JMS, ils remplissent la même fonction que les sessions beans, porteurs de logique applicative mais sont invoqués de manière asynchrone et sans mémoire de session (stateless).
La figure 11.3 montre également par quels protocoles les clients peuvent invoquer les EJB : essentiellement RMI-IIOP pour les clients Java, CORBA-IIOP pour les clients CORBA, le protocole JMS pour les clients MOM (Messaging client sur la figure) et HyperText Transfer Protocol (HTTP) pour les clients légers. J2EE est donc fortement interopérable grâce au support des protocoles IIOP et HTTP.

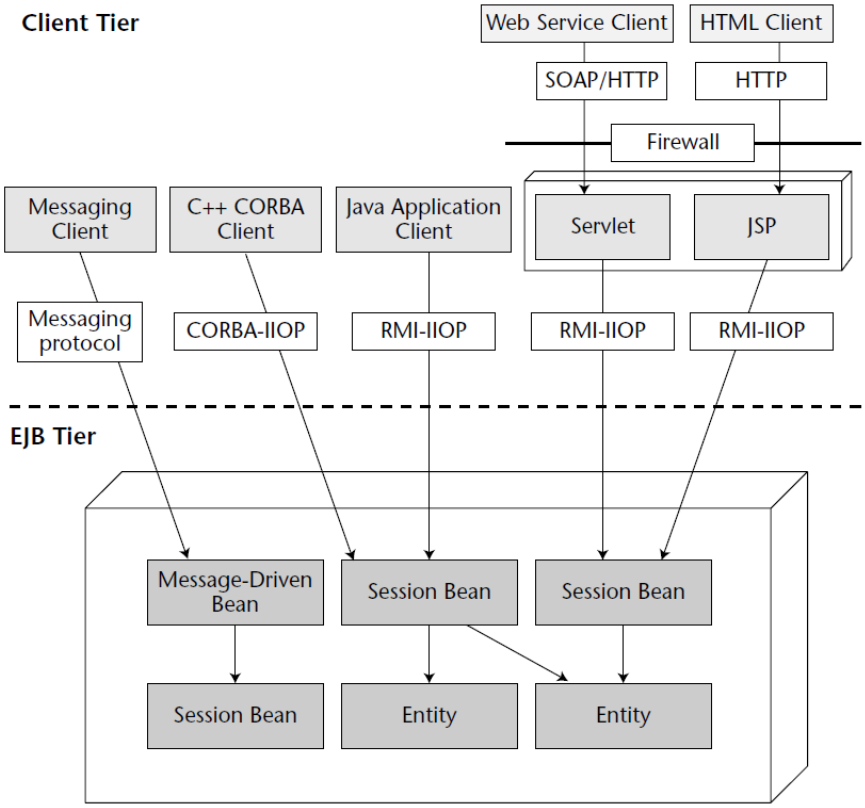

FIG. 11.3 : Schéma des composants du container EJB en interaction avec les différents clients (d'après SRIGANSEH *et al.*, 2006).

11.3.1 Choix de déploiement

L'équipe a choisi l'environnement J2EE JBoss 4.2 (Red Hat) implémentant la norme EBJ 3.0 récemment sortie à l'époque du développement et qui apporta une appréciable simplification dans la conception des EJB. JBoss est distribué sous licence LGPL et est connectable aux bases de l'IRD. C'est un environnement robuste, implémentant l'ensemble des services J2EE. Il embarque la bibliothèque Hibernate concernant la persistance des objets en bases de données relationnelles et Tomcat comme container de servlets et JSP. Le développement des modules du serveur a bénéficié de l'environnement de programmation Netbeans (Sun Microsystems, licence GPL) pour générer l'emballage de la couche applicative (Enterprise ARchive ou EAR) du serveur de modélisation sous JBoss.

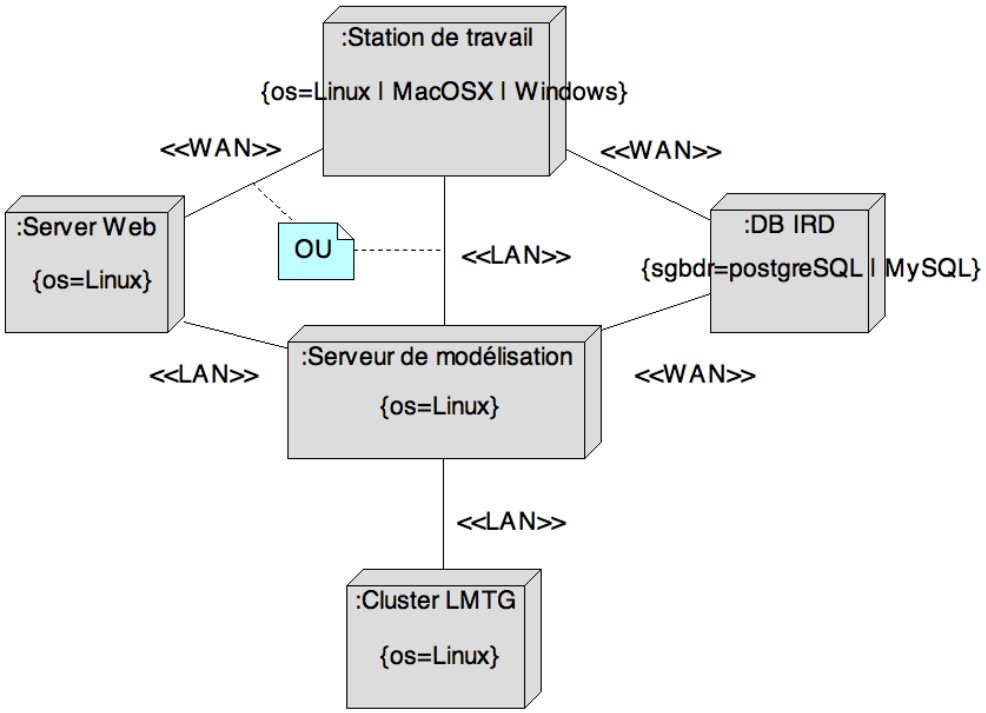

FIG. 11.4 : Diagramme de déploiement d'Umodelis.

La figure 11.4 représente le diagramme de déploiement de la plateforme Umodelis. Deux points sont importants : le lien entre le client et les bases de données est conservé pour ne pas refuser aux modules l'accès à ces dernières et une connexion entre le serveur de modélisation et la ferme de calcul du LMTG est envisagée. Les deux modes de connexion au serveur de modélisation sont également représentés : soit par réseau local (Local Area Network ou LAN avec RMI-IIOP) pour les clients lourds, soit au travers du Web (Wide Area Network ou WAN avec HTTP) pour les clients Web.
Les sections suivantes traitent de l'intégration des fonctionnalités du serveur de modélisation d'Umodelis dans l'architecture J2EE.

11.4 Accès aux données hydrologiques

Le module de persistance et d'accès aux données hydrologiques du serveur de modélisation, réalisé par l'équipe, assure les fonctions de persistance de données et leur encapsulation dans les structures de la bibliothèque d'objets hydrologiques. Il s'appuie directement sur le service de connexion au EIS proposé par l'architecture J2EE : les classes d'objets hydrologiques sont transformées en entity beans par injection d'annotations EJB3. L'injection d'annotations permet de lier une classe avec une table de base de données, ses propriétés avec les colonnes de la table.
La représentation des données en base est donc assurée par le container d'EJB qui se charge également des opérations de persistance. L'extraction des objets repose également sur le service proposé par J2EE par l'injection de requêtes nommées (named queries) dans les classes des objets hydrologiques.
Le module d'accès propose un contrôleur sous la forme d'un session bean stateful nommé MuskProjectBean aux modules clients Umodelis. MuskProjectBean contient les algorithmes, basés sur les requêtes nommées, effectuant l'extraction des objets hydrologiques ainsi que leur persistance en cascade. Le choix d'un session bean stateful a été dicté par le besoin de maintenir l'état de conversation entre le serveur et le client pour effectuer les instanciations des entity beans à la demande (lazy loading) dans le souci d'optimiser la consommation de ressources du serveur mais surtout pour éviter le partage de données entre simulations (spécification énoncée au chapitre 5 page 27).

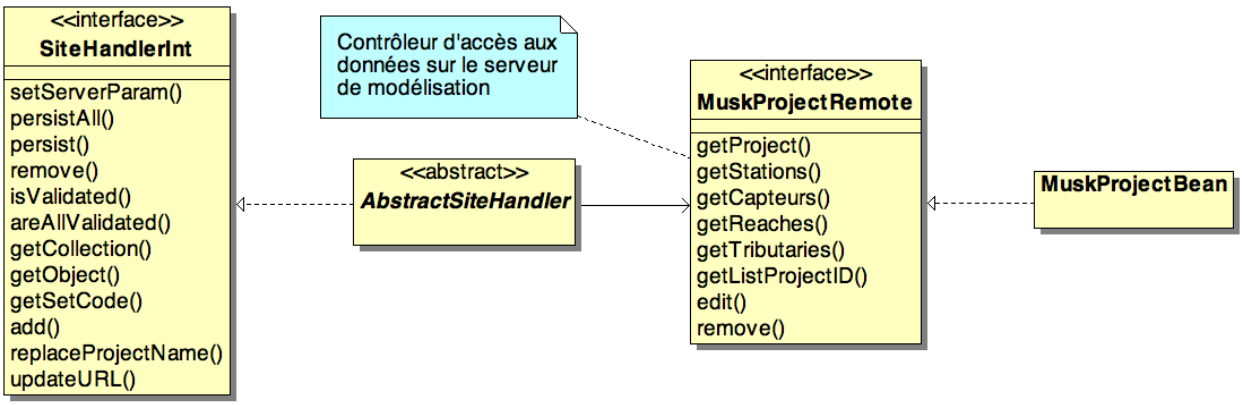

FIG. 11.5 : Extrait du diagramme de classes illustrant la relation entre le système de gestion d'objets hydrologiques et le serveur de modélisation hydrologique.

Les figures 11.5 et 11.6 qui complètent respectivement les figures 9.23 page 76 et 9.24 page 77, illustrent les relations entre, d'une part, le système de gestion de projets et d'objets hydrologiques et d'autre part MuskProjectRemote, l'interface du session bean MuskProjectBean, contrôleur du module d'accès aux données du serveur. ProjectHandlerImp et AbstractSiteHandler sont concernés par les méthodes de persistance (edit) et d'effacement (remove) générique d'objets hydrologiques et des projets (entities bean) tandis que ProjectsManager est client de la méthode de listage de projets enregistrés (getListProjectID) et des méthodes de récupération des collections d'objets hydrologiques/EJB liées à un projet particulier (getReaches, getStations, etc.).

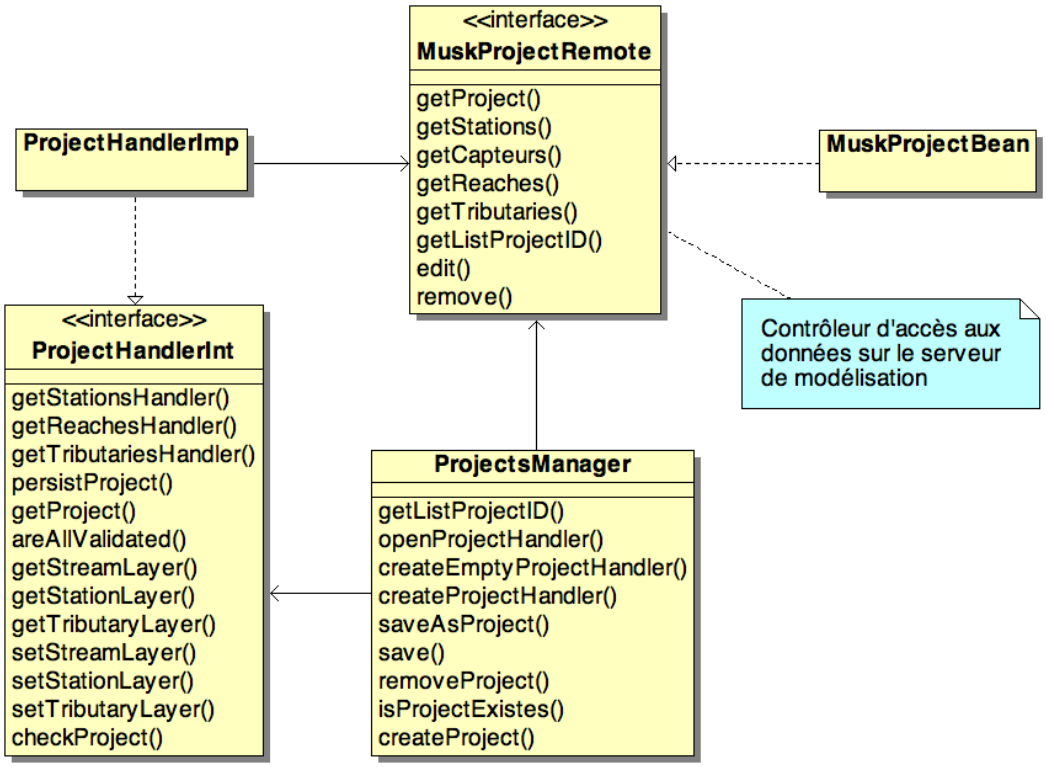

FIG. 11.6 : Extrait du diagramme de classes illustrant la relation entre le système de gestion de projets et le serveur de modélisation.

11.5 Modèles hydrologiques

Confié par l'équipe, mon travail sur les modèles hydrologiques a consisté à les intégrer au serveur de modélisation en réalisant les infrastructures nécessaires et le contrôle des simulations à distance. Cette section se termine sur ma réflexion sur l'encapsulation des modèles écrits en Fortran et C/C++.

11.5.1 Infrastructures

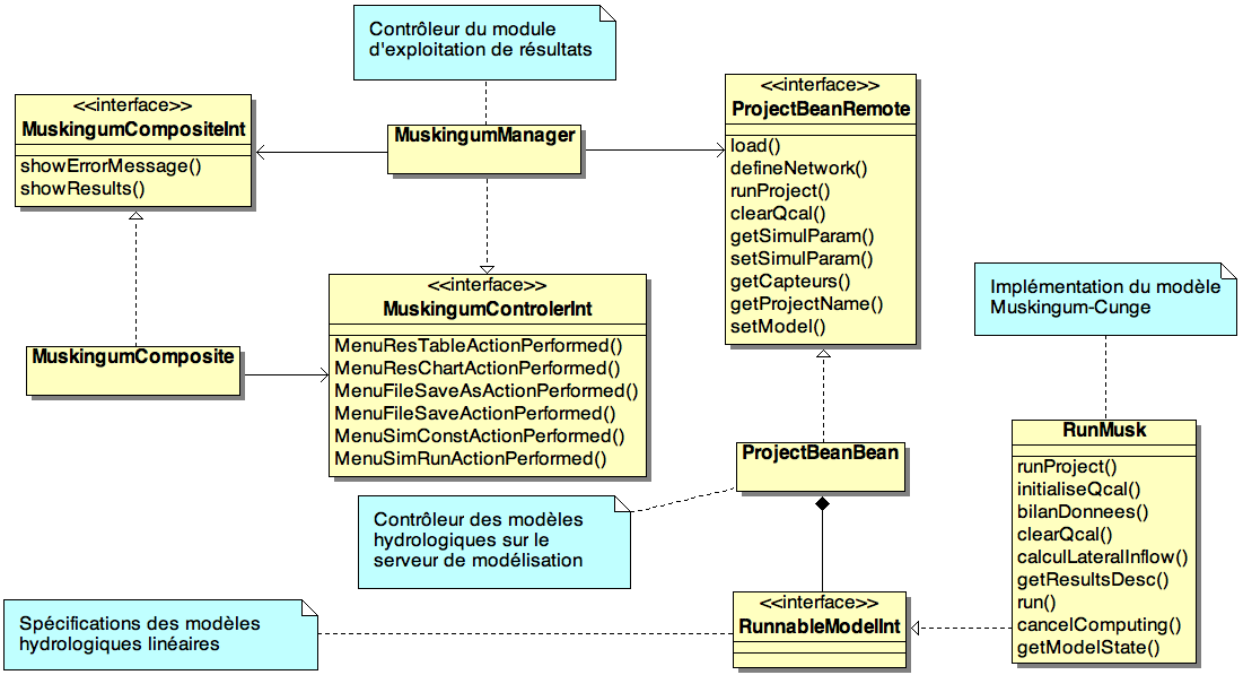

FIG. 11.7 : Extrait du diagramme de classes illustrant la relation entre le module d'exploitation de résultats et le serveur de modélisation.

L'encapsulation des modèles hydrologiques par les session beans est relativement naturelle, les modèles étant les logiques applicatives du serveur. Cependant, la multiplication de session beans de modèle favorise la redondance du code de contrôle. Afin d'encapsuler une famille de modèles hydrologiques caractérisés par les mêmes entrées/sorties, le patron de conception stratégie (design pattern strategy ; GAMMA *et al.*, 1994 ; voir annexe A.10 page 110) est tout indiqué.
Dans le cadre du module d'exploitation de résultats qui offre l'interface de contrôle des simulations des modèles hydrologiques linéaires, la classe ProjectBeanBean et son interface ProjectBeanRemote, illustrées à la figure 11.7, constituent le session bean chargé de contrôler les simulations. La classe RunMusk provenant du prototype cité au chapitre 10 page 80 fournit une implémentation du modèle Muskingum-Cunge dont la spécification est représentée par l'interface RunnableModelInt (détaillée à la section suivante). On reconnaît le patron de conception stratégie par la relation de composition sur la figure entre ProjectBeanBean et l'interface des modèles RunnableModelInt ainsi que la méthode setModel de ProjectBeanBean qui permet au client MuskingumManager (le contrôleur du module d'exploitation de résultats) de dynamiquement choisir le modèle souhaité. ProjectBeanBean centralise également les paramètres nécessaires à la simulation

et les met à disposition des modèles. A noter que le contrôle des simulations doit garder en mémoire les paramètres de simulation d'où la nature stateful du session bean.

11.5.2 Contrôleur de simulations

Son rôle est d'offrir au contrôleur du module situé du côté client de la plateforme, le moyen de superviser à distance une simulation. Comme nous l'avons précédemment vu, il ne s'agit pas seulement de choisir un modèle parmi une famille proposée mais aussi de démarrer une simulation, d'afficher un état d'avancement (par exemple sous forme de barre de progression) avec des informations (par exemple des informations textuelles et des résultats intermédiaires) sur l'opération de simulation en cours et les opérations déjà effectuées par le serveur, de rendre compte d'une erreur et d'annuler la simulation. La difficulté est de réaliser les interactions à travers le bus de communication.

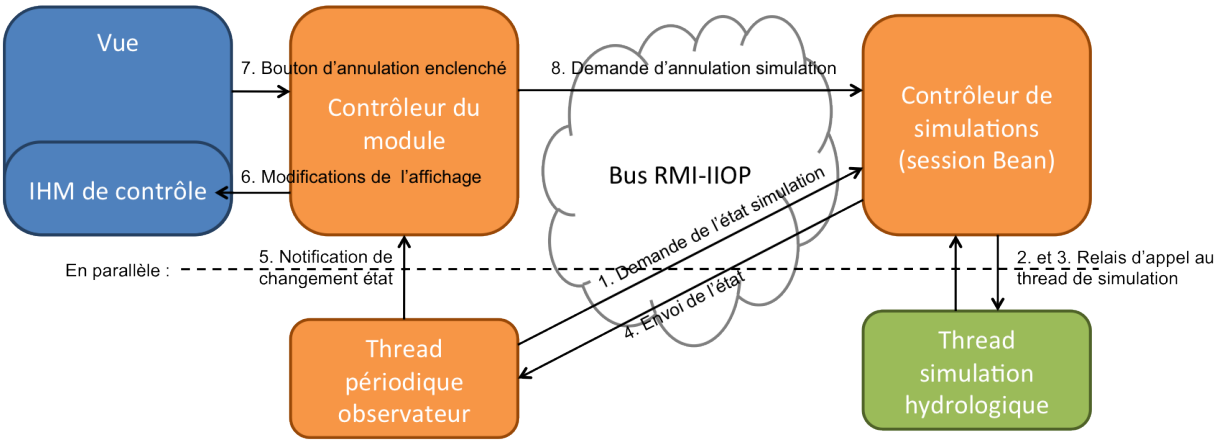

FIG. 11.8 : Schéma de flux de communications pour le contrôle des simulations hydrologiques.

L'idée directrice de la solution est de garder le client toujours disponible pour traiter les actions de l'utilisateur. Elle se traduit par la parallélisation des fils d'exécution des actions de l'utilisateur (vue et contrôleur) et celui de l'observation de la simulation, tout en partageant le même espace mémoire (ils agissent sur les mêmes entités). La figure 11.8 qui donne l'architecture mise en place, laisse apparaître deux threads : un thread périodique côté client qui est chargé de demander l'état de la simulation (tous les flux sont synchrones) au contrôleur de simulation situé côté serveur lequel relaie la demande au thread effectuant la simulation (étapes 1 à 4 sur la figure). Tandis que les contrôleurs du module et de la simulation sont disponibles pour traiter les actions de l'utilisateur (par exemple étapes 7 et 8). La modification de l'affichage de la vue reprend le patron de conception observateur : le thread périodique notifie les changements d'état de la simulation au contrôleur du module qui est chargé de modifier l'affichage de l'IHM par le biais des méthodes proposées par la vue (étapes 5 et 6 sur la figure).
Les inconvénients majeurs de cette conception viennent de la concurrence introduite entre les deux fils d'exécution côté client (les étapes 1 et 8 sont concurrentes) qui est cependant gérée par le container d'EJB. D'autre part, aucun autre client ne peut superviser la même simulation du fait de l'exclusivité du client sur le contrôleur de simulation (session bean stateful). Une autre approche architecturale est proposée dans la section alternative avec la diffusion (broadcast) de messages asynchrones à l'aide de JMS.

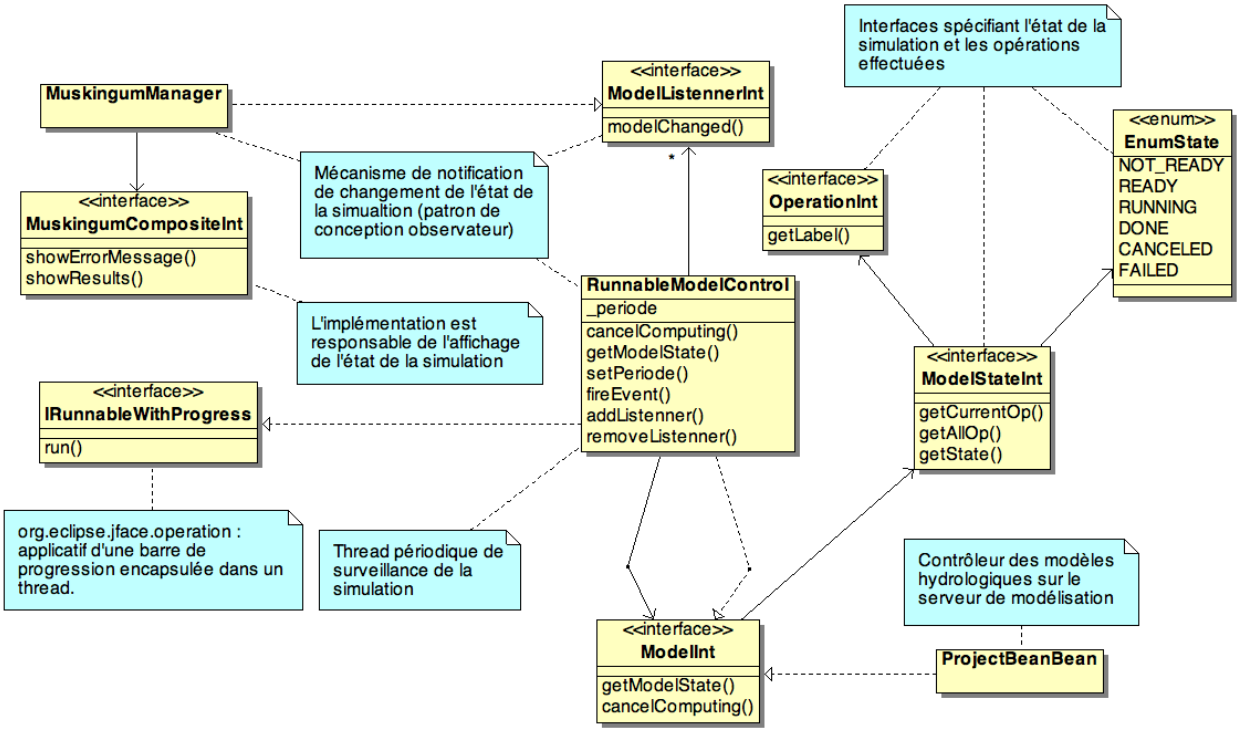

FIG. 11.9 : Extrait du diagramme de classes du contrôle de simulations hydrologiques côté client.

Du point de vue structurel, l'architecture côté client se traduit par l'apparition de la classe RunnableModelControl (figure 11.9) qui modélise le thread périodique observateur de la simulation. Comme la figure 11.9 le montre, le thread est associé indirectement au contrôleur de simulation ProjectBeanBean par l'une de ses interfaces, ModelInt qui offre les méthodes de récupération de l'état de simulation. La notification des changements d'état de la simulation est organisée selon le patron de conception observateur formé par la RunnableModelControl, ModelListenerInt et MuskingumManager. RunnableModelControl est la classe observée qui implémente les méthodes d'ajout et de suppression d'observateurs addListenner et removeListenner. ModelListenerInt est l'interface des observateurs et MuskingumManager, le contrôleur du module qui réalise cette interface, est chargé de modifier l'IHM du module grâce aux méthodes exposées par MuskingumCompositeInt. L'état de la simulation est spécifié par l'interface ModelStateInt associée à EnumState, une énumération des états possibles des simulations. OperationInt est l'interface des objets encapsulant les informations sur les opérations effectuées par la simulation.
Concernant l'IHM de la barre de progression (différente de l'IHM de la vue), elle est fournie par la bibliothèque d'Eclipse RCP (portable pour tous les environnements Java) et encapsule dans un thread la classe RunnableModelControl qui lui sert d'applicatif au moyen de l'interface IRunnableWithProgress.

La barre de progression capture les actions de l'utilisateur pendant la durée de la simulation et possède un bouton d'annulation de la simulation. La pression de ce bouton déclenche l'exécution sur le fil d'exécution principal (thread SWT) des traitements d'annulation qui sont situés dans la classe RunnableModelControl pour des raisons de centralisation. RunnableModelControl réalise la même interface que le contrôleur de simulation du serveur afin de rendre les mêmes services, annulation dans notre cas, mais du côté client.

RunnableModelControl est donc un mandataire de ProjectBeanBean.

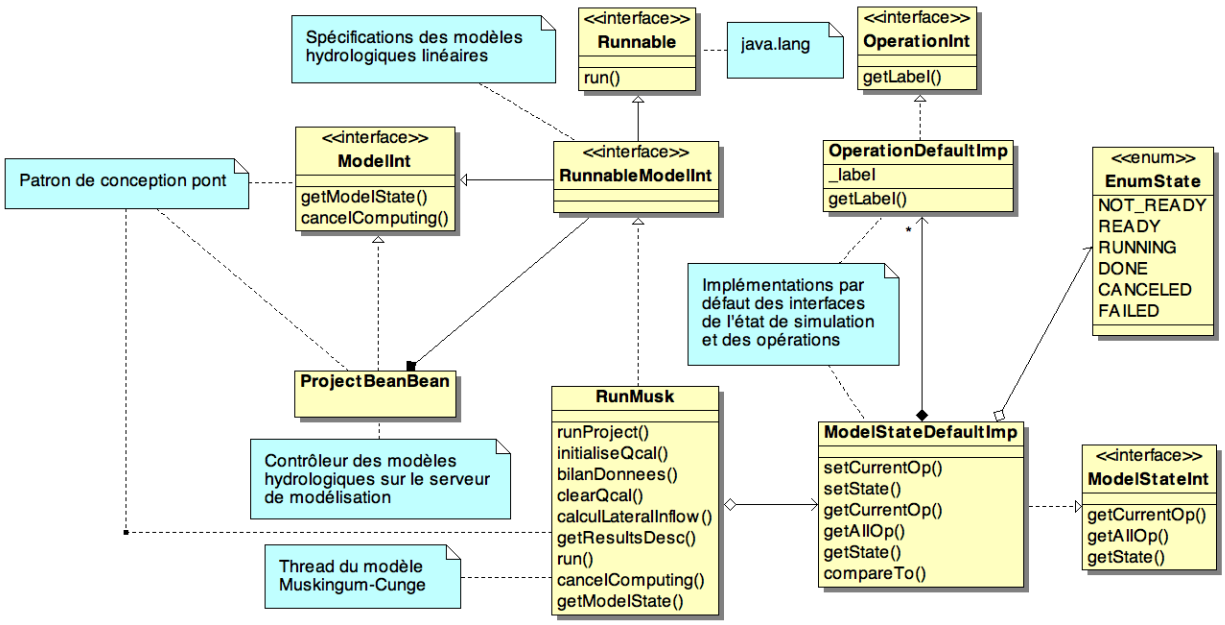

FIG. 11.10 : Extrait du diagramme de classes du contrôle des simulations hydrologiques côté serveur.

Côté serveur, la figure 11.10 fait apparaître le patron des modèles hydrologiques (RunnableModelInt) qui spécifie qu'un modèle (RunMusk) doit présenter des méthodes de contrôle (ModelInt) et constituer l'applicatif d'un thread (Runnable).
La figure 11.10 montre également les implémentations des interfaces modélisant l'état de la simulation (OperationDefaultImp et ModelStateDefaultImp) qui sont proposées par défaut aux modèles hydrologiques, libre aux modèles d'utiliser d'autres implémentations plus adaptées.
Enfin, ProjectBeanBean et RunMusk forment avec ModelInt le patron de conception pont (design pattern bridge ; GAMMA *et al.*, 1994 ; voir annexe A.9 page 109) afin de relayer les appels de contrôle ou d'observation reçus par ProjectBeanBean vers RunMusk.
Un diagramme de séquences montrant le scénario décrit par la figure 11.8 est donné en annexe B page 111.

11.5.3 Alternative avec JMS

Cette alternative architecturale cherche à rectifier les défauts de l'architecture de contrôle de simulations actuellement implémentée (appels concurrentiels et mono superviseur). Ce travail est basé sur le système de messagerie JMS dont les spécifications sont données par Sun Microsystems et dont une implémentation est proposée par JBoss (JBoss Messaging). La nouvelle architecture est calquée sur les échanges de messages asynchrones de type publication/abonnement, illustrés sur la figure 11.11 : un unique rédacteur alimente une file de messages d'un sujet et des clients s'abonnent au sujet afin d'en recevoir les messages de façon asynchrone et cohérente (transaction).
La figure 11.12 présente l'architecture intégrant JMS : l'état de la simulation devient un sujet, le contrôleur de simulations est le rédacteur et le contrôleur du module, qui est un lecteur, reçoit les états de la simulation de façon asynchrone et passive (le thread périodique n'est plus utilisé). Les autres flux restent synchrones.

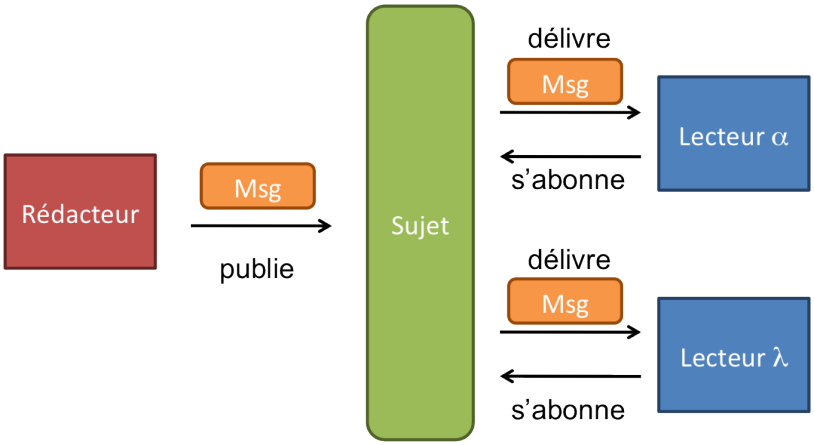

FIG. 11.11 : Schéma d'échanges de messages sur le modèle publication/abonnement.

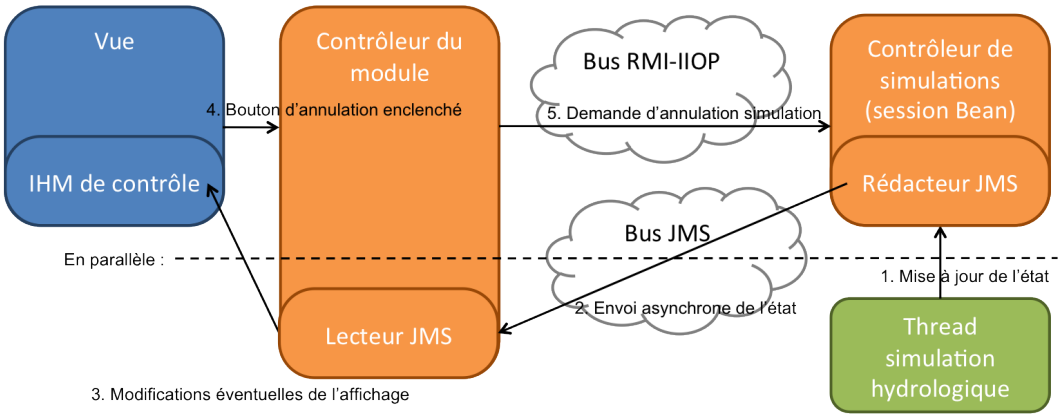

FIG. 11.12 : Schéma de flux de communications pour le contrôle de simulations hydrologiques intégrant JMS.

L'exécution du traitement des messages implémenté par le contrôleur du module est gérée par JMS dans un fil d'exécution à part (mécanisme basé sur le patron de conception observateur). A l'aide de JMS, les observations ne sont plus en concurrence avec le flux de contrôle de la simulation et l'utilisation de plusieurs clients est possible par souscription aux observations.

11.5.4 Encapsulation des modèles

Nous l'avons vu lors de la précédente section, les implémentations des modèles hydrologiques linéaires sont spécifiées par l'interface RunnableModelInt (figure 11.10). Or ces spécifications sont écrites pour du code en langage Java alors que la majorité des modèles hydrologiques sont écrits en Fortran. La solution réside dans l'utilisation de la bibliothèque Java Native Interface (JNI ; LIANG, 1999) qui permet le lien entre la JVM qui interprète les applications Java et les librairies et applications natives du système d'exploitation comme le montre la figure 11.13.

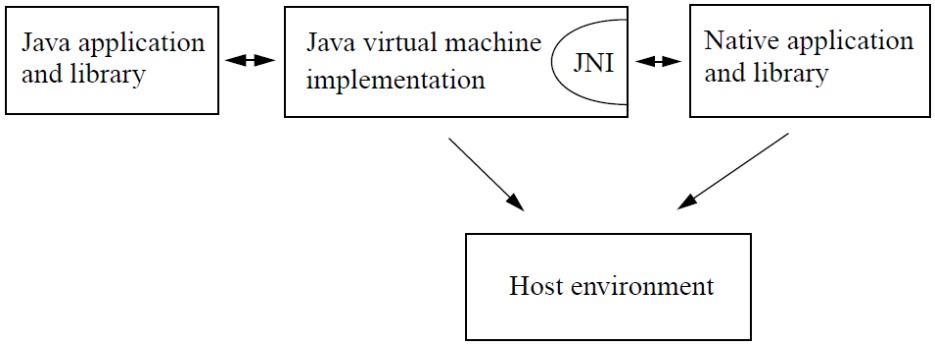

FIG. 11.13 : Rôle de JNI (d'après LIANG, 1999).

Les langages supportés par JNI sont le C et le C++. Sachant que le Fortran est depuis longtemps interopérable avec le C (afin de facilité sa migration vers le C) et a fait l'objet d'une normalisation, ISO (2003), la plateforme est capable d'encapsuler les algorithmes de modélisation écrit en Fortran et en C/C++ au prix de l'utilisation de JNI et d'une réorganisation du code des algorithmes sous forme de fonctions. Afin de faciliter la tâche, il existe des outils de génération de code Java pour l'appel de code C/C++ comme JNative. Bien qu'enfreignant la contrainte de portabilité, l'intégration de modèles hydrologiques compilés pour un système d'exploitation donné, n'a que peu d'impact puisqu'ils sont exécutés sur le serveur de modélisation : l'équipe de développement ayant le contrôle de la migration du serveur vers un autre système d'exploitation, l'exécution d'un simple script de compilation portable peut assurer le redéploiement des modèles s'ils sont écrits à l'aide de langages normalisés.

11.6 Gestion des ressources de calcul

Cette section présente ma réflexion sur la gestion des ressources de calcul par le serveur de modélisation. L'architecture du serveur de modélisation prévoit la répartition (distribution) des calculs des algorithmes vers des Unités de Calcul (UC). Afin de lever toute ambiguïté sur le terme répartition, il est important de rappeler les notions de calculs parallèles et de calculs répartis. Si leur but est de réduire le temps de

calcul et de maximiser l'exploitation des ressources de calcul, ces notions sont différentes mais néanmoins complémentaires :

11.6.1 Calculs parallèles

Le parallélisme, qui nécessiterait un traitement beaucoup plus approfondi que cette présentation, cherche à faire coopérer N UC pour effectuer un calcul. Il repose sur une architecture matérielle (SIMD et MIMD) spécifique, sur un système d'exploitation et ordonnanceur adaptés et une architecture logicielle orientée (OpenMP, MPI, etc.). Le parallélisme s'inscrit dans un contexte essentiellement mono utilisateur : un utilisateur effectue des calculs en utilisant toutes les ressources de calcul disponibles.

11.6.2 Calculs répartis (distribués)

La répartition agit comme un aiguillage de calculs indépendants les uns des autres vers des UC. L'exécution d'une série de calculs de conditions initiales différentes ou la division d'un calcul en sous calculs indépendants sur plusieurs UC entre dans la catégorie de la répartition de calculs. La répartition de calculs consiste à organiser le partage de la ressource de calcul. Le contexte est généralement multi utilisateurs.

Le parallélisme est très coûteux (acquisition de compétences, temps de développement, équipement et maintenance) et fait appel à des compétences très pointues. Par contre, la répartition est plus facile à mettre en place. Dans le cadre de la recherche hydrologique à l'IRD Brésil, le parallélisme des algorithmes n'est pas implémenté par les chercheurs dont la stratégie est de mener N simulations hydrologiques indépendantes (conditions initiales différentes) simultanément sur N UC plutôt que d'exécuter parallèlement sur N UC, les simulations les unes après les autres. Cette stratégie est adaptée puisque l'ordre du temps de simulation pour l'instant ne dépasse pas l'heure, mais la modélisation hydrologique, comme toute modélisation, atteindra un temps de simulation critique, qui nécessitera le changement de stratégie.

11.6.3 J2EE et la répartition

J2EE qui n'est pas une plateforme orientée calculs scientifiques mais une plateforme d'applications (business services), propose deux mécanismes de répartition : la balance de charge (load balancing) et la bascule d'urgence (failover).
Comme illustrée à la figure 11.14, la balance de charge effectue la répartition des requêtes provenant des clients J2EE vers les UC dont la charge de calcul n'atteint pas un seuil critique donné. J2EE est conçu pour assurer le traitement de multiples requêtes par exemple de type Web. Concernant Umodelis, la balance de charge serait une stratégie de répartition des simulations entre les UC de la ferme de calcul du LMTG.
Une décomposition des algorithmes de modélisation en calculs indépendants est envisageable afin de répartir plus finement les calculs sur les UC à condition que le surcoût de communication inter UC sur le bus J2EE soit justifié.
Le mécanisme de bascule d'urgence proposé par J2EE (figure 11.15) participe à la robustesse du système : en cas de panne d'une UC, une autre prend le relais afin de terminer le calcul en cours sans déni de service. Il suppose la répartition de la mémoire des UC. Son utilisation ne serait pas pertinente concernant Umodelis, la disponibilité du service de modélisation n'est pas critique et la relative faible occurence des pannes ne justifie pas le surcroît de consommation de ressources.

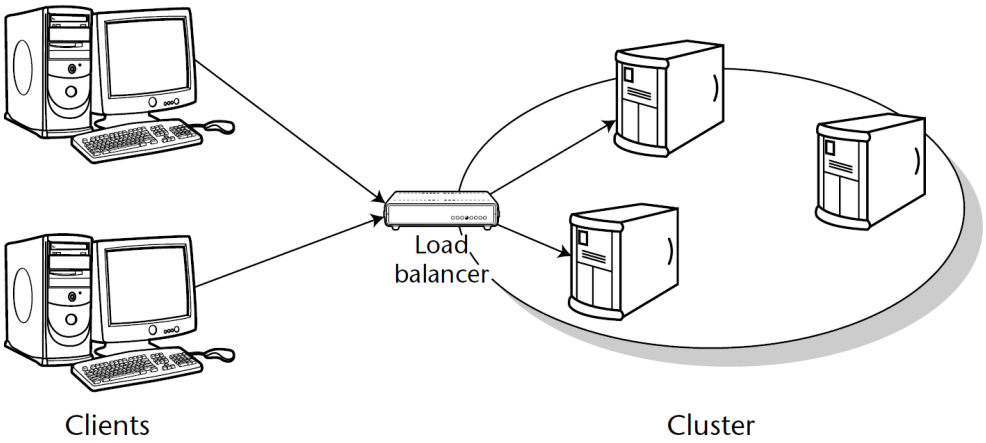

FIG. 11.14 : Schéma de balance de charge (d'après SRIGANSEH *et al.*, 2006).

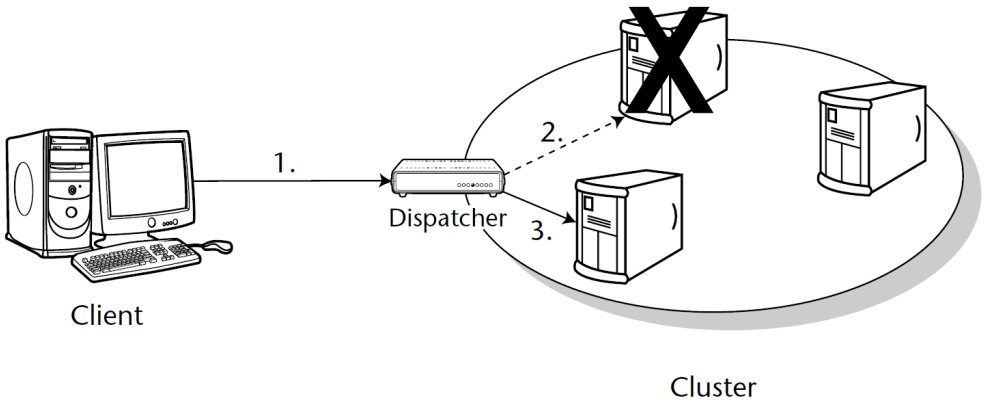

FIG. 11.15 : Schéma de basculement d'urgence (d'après SRIGANSEH *et al.*, 2006).

11.7 Conclusion

J2EE apporte une solution technique robuste, pérenne, capable de gérer la répartition de la charge de calcul et pertinente vis-à-vis des spécifications de l'architecture d'Umodelis afin d'intégrer les modules du serveur de modélisation. Ces derniers apportent au système Umodelis une gestion transparente et distribuée des objets hydrologiques stockés dans les bases de données de l'IRD, ils permettent une intégration du code des modèles hydrologiques même écrits en langages autres que Java et offrent un contrôle des simulations hydrologiques à distance.

CHAPITRE 12

Bilan et conclusion

12.1 Bilan

Au terme de ma contribution au projet, plusieurs points de la plateforme restent perfectibles. Ce chapitre propose un bilan sur la conception de la plateforme Umodelis durant la période de mon stage. Il s'articule autour de l'état du projet puis d'une discussion autocritique sur certains choix de conception.

12.1.1 État du projet

Actuellement, Umodelis se présente comme un socle de plateforme pour l'intégration d'outils de traitements et de modèles hydrologiques, telle qu'elle a été définie à la section 4.2 page 21. Bien qu'étant fonctionnelle, la plateforme n'offre que peu de fonctionnalités. Mais, le travail effectué propose une base, une architecture, où pourront se greffer d'autres travaux afin de converger vers l'outil attendu. Le côté pédagogique de la plateforme est exploitable. Certaines améliorations sont déjà prévues telles que la réorganisation de la conception du contrôleur du module d'exploitation de résultats autour du patron de conception état et un éventuel couplage avec le logiciel R afin d'offrir des fonctions statistiques avancées. Un audit du code de la plateforme est également prévu afin de consolider, par exemple, les règles de nommage ainsi que l'implémentation du contrôle de simulations à l'aide de JMS. Les futurs développements d'Umodelis devront s'orienter vers l'intégration de modèles écrits en Fortran, l'intégration de l'outil Hydraccess de l'IRD et surtout une phase de déploiement et de tests auprès des utilisateurs, qui n'a malheureusement pas été faite par manque de temps. Faute de disponibilité du personnel et de moyens, le projet est d'ailleurs contraint à l'arrêt.

12.1.2 Discussion des choix de conception

Cette section aborde du point de vue critique le choix d'uDig, la conception du découplage entre Umodelis et le client SIG, l'intégration de modèles hydrologiques et la conception du serveur de modélisation.

12.1.2.1 uDig

Le choix de travailler sur une version candidate d'uDig 1.1 (section 6.2.3.2, page 38), bien qu'inévitable pour l'avenir de la plateforme, a relativement freiné le développement d'Umodelis. La résolution des bugs ou leur contournement a couté un temps non négligeable, si bien que le développement touchant au couplage entre Umodelis et uDig fut au détriment de l'intégration d'autres modèles (notamment ceux écrit en Fortran). Néanmoins, la plus value la plus importante d'Umodelis est la représentation des données de modèles par un SIG : sur ce point, la plateforme est avancée. D'autre part, le choix de l'implémentation de la plateforme en Java, lié au choix d'uDig et à l'expertise technique de l'équipe de développement, limite

la plateforme aux SIG offrant des points d'extension dans ce langage. Le serveur de modélisation n'est pas touché par cette restriction car il est accessible par les protocoles RMI-IIOP et HTTP.

12.1.2.2 Découplage d'Umodelis et du SIG

Le couplage faible entre Umodelis et le SIG est relativement bien conçu, l'isolation du SIG est garantie. Mais il est au prix d'une certaine lourdeur introduite par l'utilisation de plusieurs patrons de conception souvent imbriqués les uns dans les autres. D'une part, le changement de SIG (ou la modification de la bibliothèque fournie par uDig) oblige à implémenter un certain nombre d'interfaces de la bibliothèque SIG, ce qui augmente le volume de classes et donc la complexité du logiciel. D'autre part, la sémantique des méthodes est figée. Toutes les classes ou les méthodes des bibliothèques fournies par les SIG ne sont probablement pas factorisables en méthodes génériques communes à tous les SIG. Le risque de concevoir des méthodes spécifiques à un SIG en particulier est grand. Dans l'état actuel du projet, toutes les méthodes sont génériques (études préliminaires sur uDig et ArcGIS). Heureusement, le changement de SIG n'est pas fréquent.

12.1.2.3 L'intégration des modèles

L'intégration de modèles écrits en code Fortran est un pari ambitieux dont la faisabilité, gouvernée par les limitations de JNI, reste à prouver. Nul doute que le code des chercheurs sera à réorganiser profondément (le développement n'est pas le cœur de métier des chercheurs en sciences de la Terre). La retro conception du code des modèles ne va pas à l'encontre des contraintes imposées au projet mais génèrera un travail important.

12.1.2.4 Serveur de modélisation et architecture J2EE

J2EE est une architecture facile à mettre en œuvre ; toutefois, sa pertinence, à long terme, pour les calculs scientifiques, n'est pas évidente. Evoquée dans la section 11.6.2 page 97, la parallélisation, dans un futur plus ou moins proche, des modèles hydrologiques sera nécessaire. Le problème soulevé est la compatibilité entre le serveur de modélisation et les modèles parallélisés. Des pistes existent : pour les modèles écrits en FORTRAN ou en C, la faisabilité est encore une fois déterminée par JNI. Pour les modèles écrits en Java, l'API ProActive (licence GNU), issue des travaux de l'INRIA, propose une parallélisation sur le modèle de Single Process, Multiple Data (SPMD). Le nombre de clients se connectant simultanément au serveur de modélisation ne devrait pas dépasser la dizaine. Il est donc peu probable que la gestion des clients par le serveur pose un problème. Par contre, le volume de données transitant sur le réseau est problématique surtout pour des connexions dont la bande passante est réduite. Le serveur de modélisation devra être connecté sur une ligne large bande et réactive afin de ne pas pénaliser les clients pour le surcroit de latence dû à la liaison entre les bases de données et le serveur. La liaison client-serveur devra payer un surcoût non négligeable dû à la sérialisation des informations structurelles des objets hydrologiques. L'utilisation de caches côté client et serveur est envisageable pour limiter le transport de données mais au prix, parfois lourd, de la gestion de ces caches.

12.2 Conclusion

Les grands bassins versants ont une influence importante sur la dynamique globale de la Terre. Leur étude est une priorité pour l'IRD et leurs partenaires. Plusieurs disciplines scientifiques ont ainsi pour objet les bassins versants et partagent un certain nombre de données, d'outils de traitements et de modèles mathématiques. Le projet Umodelis propose une base de plateforme, qui répond au besoin de fédérer les outils de traitements et les modèles mathématiques, notamment ceux appliqués en hydrologie, afin d'exploiter leur complémentarité par chaînage d'outils ou de les comparer. Cette base offre en outre un cadre pour la distribution des modèles et des traitements hydrologiques afin de partager la connaissance scientifique et les ressources de calcul mises à disposition des utilisateurs.

La plateforme uniformise la source et la nature des données, résout l'hétérogénéité des métadonnées les décrivant afin de faciliter leur accès distribué. Elle répond également au besoin de faciliter la mise en forme des données en entrée des outils de traitements notamment grâce à l'intégration d'un SIG. Elle profite des fonctionnalités du SIG concernant la gestion et la représentation graphique de l'information spatialisée provenant de diverses sources, locales ou distantes.

L'architecture d'Umodelis reprend le schéma classique des architectures n tiers. Elle est composée de deux sortes de clients qui accèdent aux outils de traitements : un client Web et un client SIG qui possède en plus une couche graphique capable de gérer et de représenter graphiquement les objets géographiques. Les clients communiquent avec le serveur de modélisation hydrologique, qui emballe les données stockées dans les bases de données de l'IRD proposant ainsi une gestion plus simplifiée. Le serveur intègre les modèles mathématiques issus des travaux de recherche et offre également une interface de contrôle à distance du déroulement des simulations dont les calculs sont potentiellement distribuables grâce à l'architecture J2EE.

La nature fortement orientée composant de l'architecture assure l'extension de la plateforme par ajout de modules respectant les spécifications fournies. La bibliothèque des données géomatiques est conçue sur le même principe, facilitant l'intégration des concepts nécessaires aux modèles.
Umodelis a été conçue en minimisant ses dépendances extérieures et particulièrement celle entre la plateforme et le SIG : la bibliothèque SIG propose des interfaces et des mécanismes basés sur les patrons de conception réutilisables afin de réaliser un couplage faible.

Umodelis livre pour le moment deux modules de traitements intégrés dans les clients : le module de création de sites hydrologiques et le module d'exploitation de résultats. Le module de création de sites hydrologiques permet d'extraire des biefs de cours d'eau et leurs stations hydrologiques afin de les associer à des données hydrologiques spatialisées ou non qui serviront d'entrées aux modèles hydrologiques linaires. Le module d'exploitation de résultats offre l'interface de contrôle des simulations et l'interface de dépouillement des résultats sous forme de tableaux ou de graphiques et constitue un exemple d'intégration d'un outil autonome non conçu pour la plateforme.

Côté serveur de modélisation, Umodelis implémente les composants d'accès et de persistance des données, le patron des modèles hydrologiques linaires contrôlables à distance dont une implémentation du modèle Muskingum-Cunge, pour le calcul du débit d'un cours d'eau par propagation, est donnée en exemple.
Enfin, les composants du système Umodelis sont écrits en Java assurant sa portabilité sur les systèmes d'ex-

ploitation les plus utilisés. La plateforme n'utilise que des environnements distribués sous licence LGPL ou équivalentes et promeut les standards émergeant de la géomatique actuelle.

Mon implication dans ce projet a été l'occasion de renforcer mes compétences relationnelles et techniques. Il m'a permis de mieux appréhender la gestion de projet dans le cadre d'un travail collaboratif. La conception d'Umodelis a nettement amélioré ma rigueur pour la spécification et a affiné ma compétence en langage UML au point d'en faire une habitude dans mon processus de conception et développement logiciel. D'autre part, j'ai acquis une compétence opérationnelle de l'architecture J2EE et Eclipse RCP ainsi qu'une meilleure connaissance du fonctionnement des architectures n tiers et des intergiciels. Par la même occasion, ma culture scientifique et technique s'est enrichie de deux nouveaux domaines : la modélisation hydrologique et la géomatique.

Enfin, ce stage fut une expérience humaine enrichissante et inoubliable. Il m'a permis d'habiter et de travailler dans un pays étranger, de m'immerger au sein de sa population pour mieux découvrir et comprendre sa culture.

ANNEXES

Annexe A

Patrons de conception

A.1 Adaptateur (adapter)

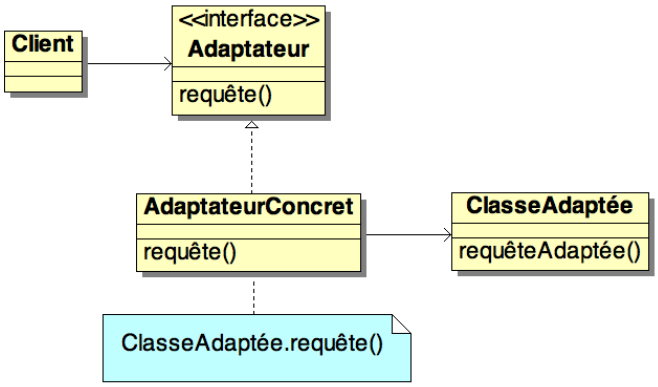

Fig. A.1 : Patron de conception adaptateur.

Convertit l'interface d'une classe en une autre conforme à l'attente du client. L'adaptateur permet à des classes de collaborer, alors qu'elles n'auraient pas pu le faire à cause d'interfaces incompatibles (figure A.1).

A.2 Chaîne de responsabilité (chain of responsability)

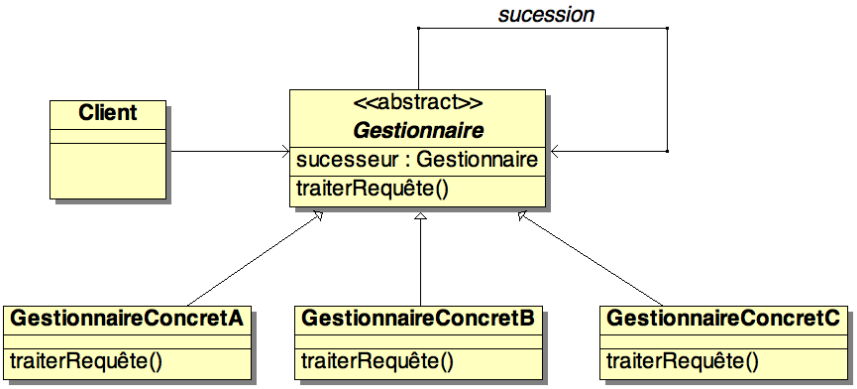

Fig. A.2 : Patron de conception chaîne de responsabilité.

Utiliser le pattern chaîne de responsabilité pour intercaler des gestionnaires dans le flux de traitement de requêtes (figure A.2).

A.3 Commande (command)

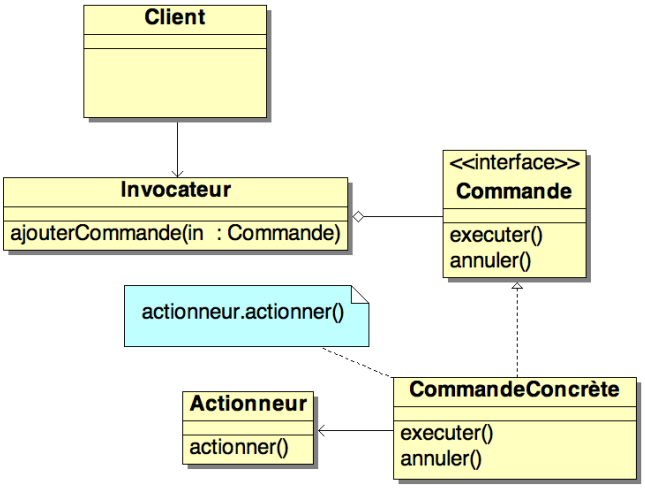

FIG. A.3 : Patron de conception commande.

Encapsule une requête comme un objet, autorisant ainsi son paramétrage par les clients, l'implémentation de files d'attente de requêtes, de récapitulatifs de requêtes et autorise la réversibilité des opérations (figure A.3).

A.4 Etat (state)

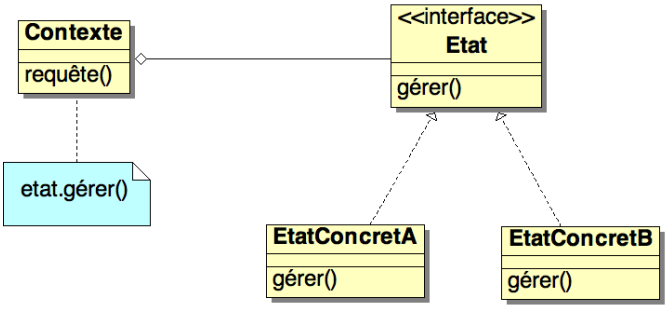

FIG. A.4 : Patron de conception état.

Permet à un objet de modifier son comportement dynamiquement quand son état interne change. Tout se passe comme si l'objet changeait de classe (figure A.4).

A.5 Fabrique (factory)

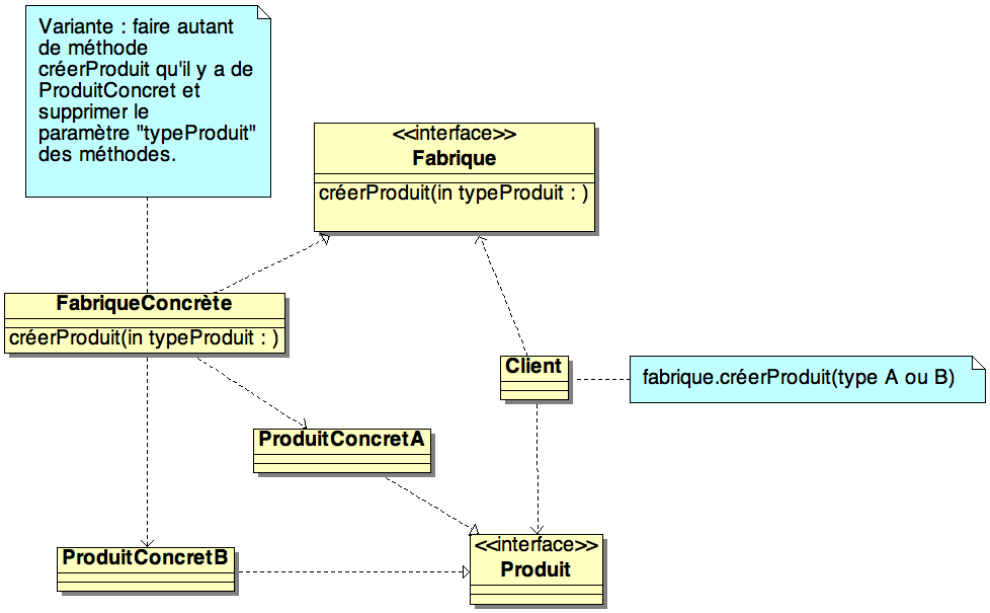

FIG. A.5 : Patron de conception fabrique.

Définit une interface pour la création d'un objet, mais en laissant aux sous-classes, le choix des classes à instancier. Fabrique permet à une classe de déléguer l'instanciation d'un objet à des sous-classes (figure A.5).

A.6 Fabrique abstraite (abstract factory)

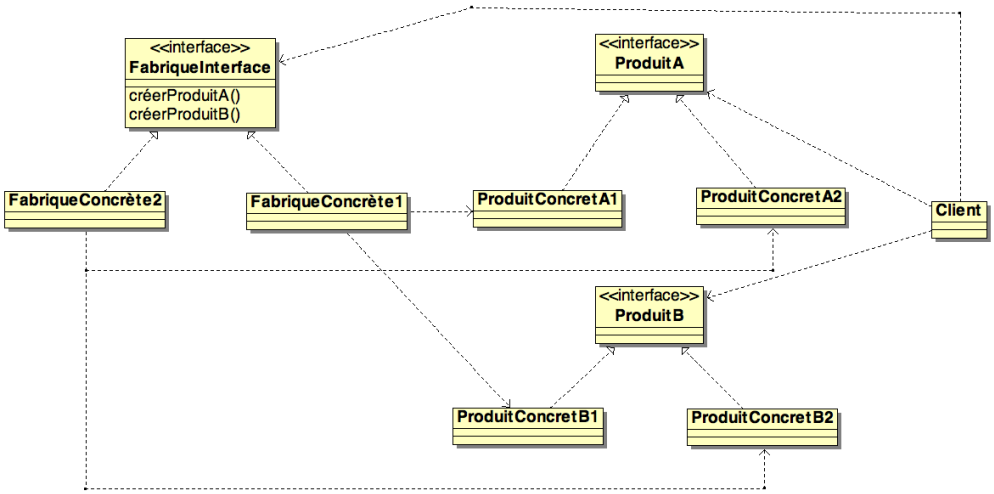

FIG. A.6 : Patron de conception fabrique abstraite.

Fournit une interface pour créer des familles d'objets apparentés ou dépendants sans dépendance vis-à-vis des classes concrètes (figure A.6).

A.7 Mandataire (proxy)

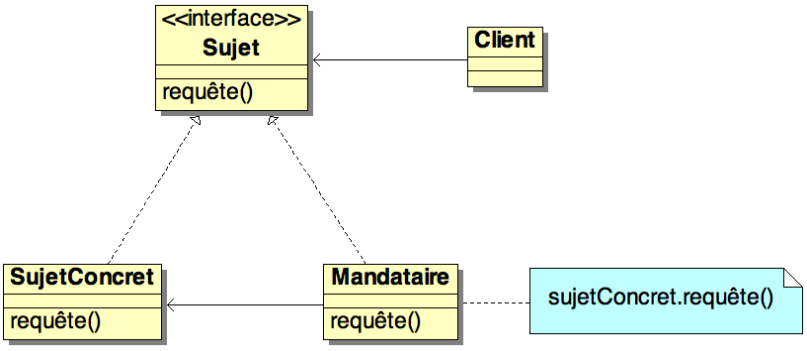

FIG. A.7 : Patron de conception mandataire.

Fournit un mandataire à un autre objet, pour en contrôler l'accès (figure A.7).

A.8 Observateur (observer)

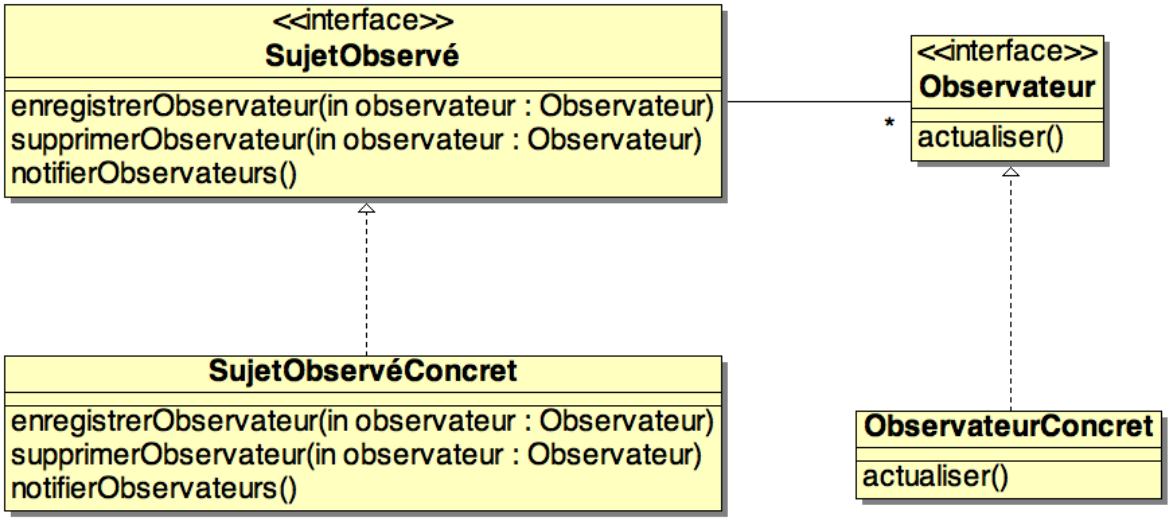

FIG. A.8 : Patron de conception observateur.

Définit une relation un à plusieurs, de façon que, lorsque le sujet d'observation change d'état, tous ceux qui l'observent en soient notifiés et mis à jour automatiquement (figure A.8).

A.9 Pont (bridge)

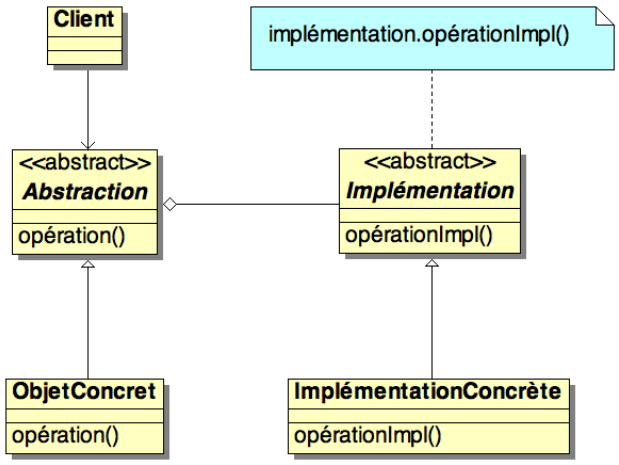

FIG. A.9 : Patron de conception pont.

Le pont est un motif de conception qui permet de découpler l'interface d'une classe et son implémentation. Ainsi l'interface et l'implémentation peuvent varier séparément (figure A.9).

A.10 Stratégie (strategy)

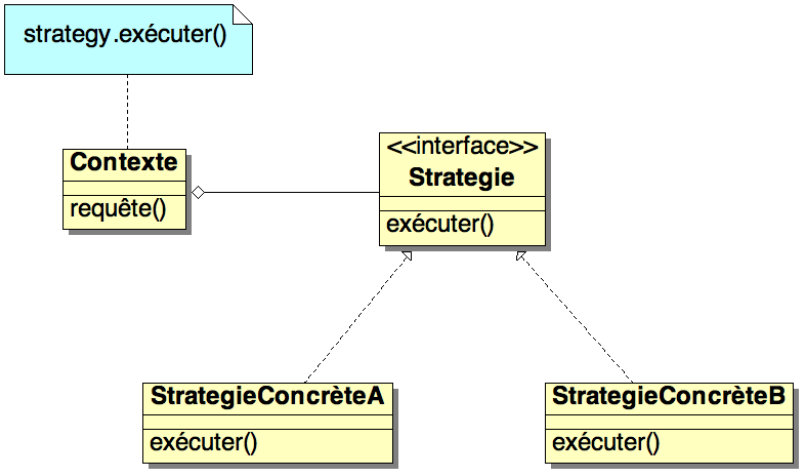

FIG. A.10 : Patron de conception stratégie.

Définit une famille d'algorithmes, encapsule chacun d'eux et les rend interchangeables. Stratégie permet à l'algorithme de varier indépendamment des clients qui l'utilisent (figure A.10).

Annexe B

Scénario de contrôle de simulations

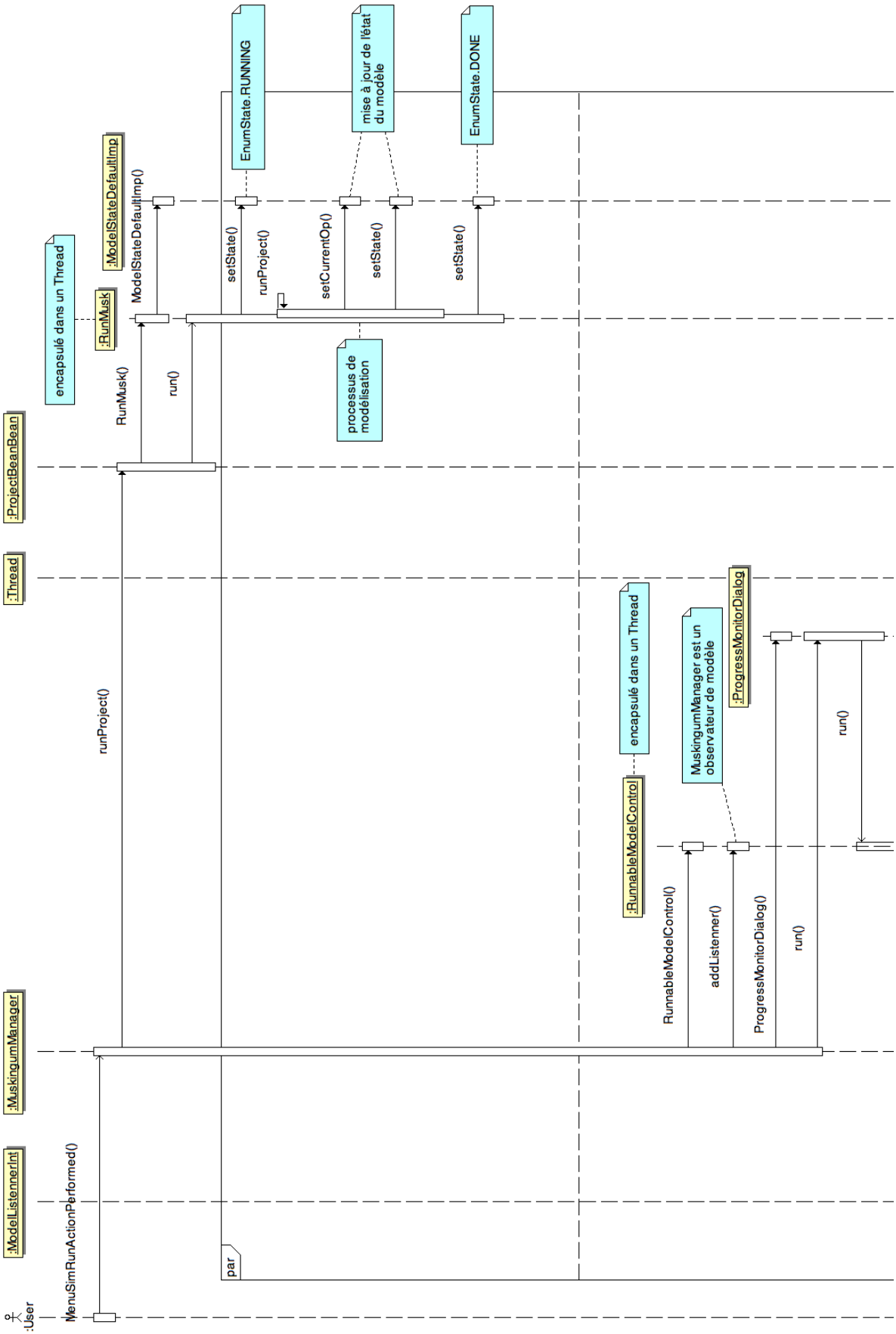

FIG. B.1 : Diagramme de séquences d'un scénario de contrôle de simulations.

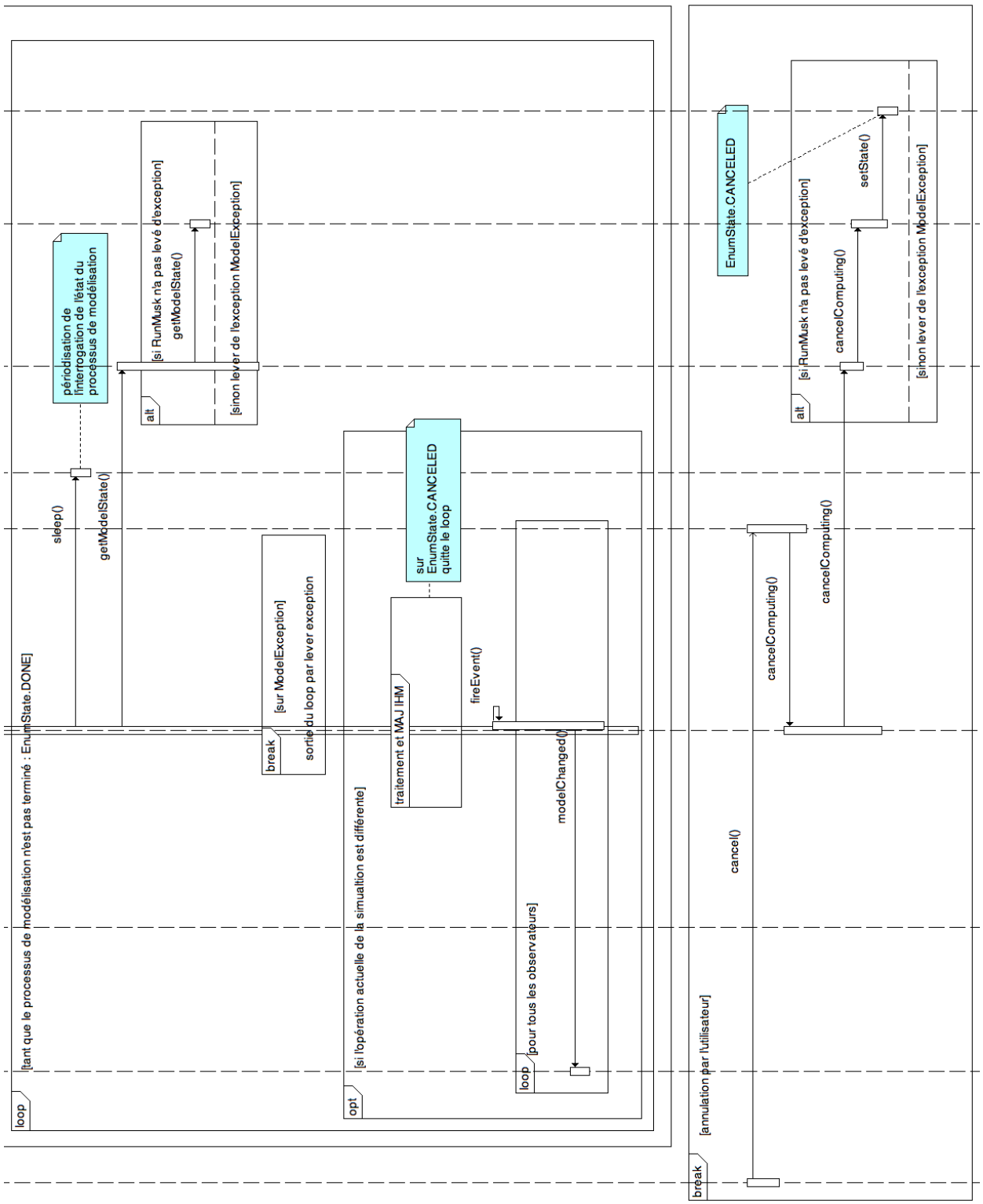

FIG. B.2 : Diagramme de séquences d'un scénario de contrôle de simulations.

Bibliographie

BAUMGARTNER A., REICHEL E., 1975, *The world water balance*, Elsevier, 179 p.

BOUVIER C., DELCAUX F., CRESPY A., 1996, ATHYS : atelier hydrologique spatialisé, *IAHS*, **238**, 425–435.

BRANGER F., 2007, Plateformes de modélisation environnementale, In *Utilisation d'une plateforme de modélisation environnementale pour représenter le rôle d'aménagements hydro-agricoles sur les flux d'eau et de pesticides*, 21–30.

CASHTEAM, 2006, Online altimetry service for hydrology : the CASH project, In *Fifteen Years of Progress in Radar Altimetry*, Venice, Italy.

CHRISTIANSEN J., 2000, A flexible object-based software framework for modeling complex systems with interacting natural and societal processes, In *4th international conference on integrating GIS and environmental modeling*, Banff, Alberta, Canada, 10.

COCHONNEAU G., GARDOLL S., BONNET M., 2008, Une plate-forme de modélisation de la variabilité spatio-temporelle des flux au sein du basin amazonien, In *13th World Water Congress*, Montpellier, France, 10.

COCHONNEAU G., SONDAG F., GUYOT J., BOAVENTURA G., FILIZOLA N., FRAIZY P., LARAQUE A., MAGAT P., MARTINEZ J., NORIEGA L., OLIVEIRA E., ORDONEZ J., POMBOSA R., SEYLER F., SIDGWICK J., VAUCHEL P., 2006, L'observatoire de recherche en environnement ORE-HYBAM sur les grands fleuves amazoniens, *IAHS*, **308**, 44–50.

CUNGE J., 1969, On the subject of a flood propagation computation method (Muskingum method), *Hydraul Res*, **7(2)**, 205–230.

DEGENS E., KEMPE S., RICHEY J., 1991, Biogeochemistry of major world rivers, In *Biogeochemistry of major world rivers, SCOPE, 42*, Degens E.T., Kempe S., Richey J.E., Eds, J. Wiley., 323–347, http://www.icsu-scope.org/downloadpubs/scope42/chapter15.html.

GAMMA E., HELM R., JOHNSON R., VLISSIDES J., 1994, *Design Patterns : Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, 416 p.

GARNETT J., 2005, User-friendly Desktop Internet GIS, In *Miles Vitual Seminar*, 7, http://udig.refractions.net/docs/VSpaperuDig.pdf.

GREGERSEN J., GIJSBERS P., WESTEN S., BLIND M., 2005, Open MI : the essential concepts and their implications for legacy software, *Advances in geosciences*, **4**, 37–44.

GUYOT J., 2005, ORE-HYBAM : contrôles géodynamique, hydrologique et biogéochimique de l'érosion / altération et des transferts de matière dans le bassin de l'Amazone, In *la Lettre du Changement global*, volume 18, PIBC-PMRC, France, 21–22.

IRD, 2007, *Rapport d'activité*, IRD, 64 p.

IRDBRESIL, 2006, *Recherches de l'IRD au Brésil depuis 1998*, IRD Brésil, 160 p.

ISO, 2003, Interoperability with C, In *Norme Fortran 2003 ISO/IEC IS 1539-1 :2004 (E)*, 391–404, http://www.oca.eu/pichon/f2k_cd.html.

KRALISCH S., KRAUSE P., 2006, JAMS a framework for natural resource model development and application, In *7th international conference on hydroinformatics*, volume 3, Nice, France, 2356–2363.

LIANG S., 1999, Introduction, In *The Java Native Interface Programmer's Guide and Specification*, 3–8, http://java.sun.com/docs/books/jni/.

MARTINEZ J., GUYOT J., COCHONNEAU G., SEYLER F., 2007, Surface Water Quality Monitoring in Large Rivers with MODIS Data, In *IEEE international Geoscience and Remote Sensing Symposium*, Barcelone, Spain, ? ? ?

MORIN G., 1991, Le modèle hydrologique CEQUEAU : exemples d'application, In *Utilisation rationnelle de l'eau des petits bassins versants en zone aride*, John Libbey Eurotext, AUPELF-UREE, Eds, 23–39.

MOUSSA R., VOLTZ M., P. A., 2002, Effects of the spatial organization of agricultural management on the hydrological behaviour of a farmed catchment during flood events, *Computer Vision and Image Understanding*, **16(2)**, 393–412.

MUSY A., 2005, Cours d'hydrologie générale de l'école polytechnique fédérale de Lausanne, http://echo.epfl.ch/e-drologie/.

RAHMAN J., SEATON S., PERRAUD J., HOTHAM H., VERRELLI D., COLEMAN J., 2003, It's TIME for a new environmental modeling framework, In *International congress on modelling and simulation*, volume 4, Townsville, Australia, 1727–1732.

RAMADE F., 1993, *Dictionnaire encyclopédique de l'écologie et des sciences de l'environnement*, Ediscience international, 822 p.

RAMSEY P., 2006, Introduction to uDig, an open source platform for GIS, In *Free and open source software for geoinformatics conference*, Lausanne, Swiss, 39, http://www.foss4g2006.org/contributionDisplay.py?contribId=132&sessionId=44&confId=1.

REENSKAUG T., 1979, THING-MODEL-VIEW-EDITOR - an Example from a planningsystem, Technical Report, Xerox PARC.

REENSKAUG T., 2003, The Model-View-Controller (MVC) Its Past and Present, In *Draft of the University of Oslo*, 16, http://heim.ifi.uio.no/trygver/2003/javazone-jaoo/MVC_pattern.pdf.

ROQUES P., VALLEE F., 2004, *UML2 en action : De l'analyse des besoins à la conception J2EE. 3ième édition*, Eyrolles, 385 p.

ROUX E., CAUHOPE M., BONNET M., CALMANT S., VAUCHEL P., SEYLER S., 2008, Daily water stage estimated from satellite altimetric data for large river basin monitoring, *Hydrological Sciences*, **53(1)**, 81–99.

ROYER A., BECKER F., GAGNON P., 2007, Trente-cinq ans d'observations spatiales de la Terre : de la photographie à l'électromagnétométrie, *Revue Télédétection*, **7**, 65–88.

SRIGANSEH R., BROSE G., SILVERMAN M., 2006, *Mastering Enterprise JavaBeans 3.0*, Wiley, 721 p.

SUN-MICROSYSTEMS, 2008, Overview, In *The Java EE 5 Tutorial*, 41–64, http://java.sun.com/javaee/reference/bookshelf/.

VAUCHEL P., 2004, Software Présentation, http://www.mpl.ird.fr/hybam/outils/hydraccess.htm.

VIALLET P., DEBIONNE S., BRAUD I., DEHOTIN J., HAVERKAMP R., SAADI Z., ANQUETIN S., BRANGER F., VARADO N., 2006, Towards multi-scale integrated hydrological models using the LIQUID framework, In *7th International Conference on Hydroinformatics*, volume 1, Nice, France, 542–549.

VILLAR J., GUYOT J., RONCHAIL J., COCHONNEAU G., FILIZOLA N., FRAIZY P., LABAT D., OLIVEIRA E., ORDOÑEZ J., VAUCHEL P., 2009, Contrasting regional discharge evolutions in the Amazon basin (1974-2004), *Journal of Hydrology*, **375**, 297–311.

VOINOV A., COSTANZA R., WAINGER L., BOUMANS R., VILLA F., MAXWELL T., VOINOV H., 1999, Patuxent landscape model : integrated ecological economic modeling of a watershed, *Environmental modeling and software*, **14**, 473–491.

YINHUAN Y., QIUMING C., 2007, Integrating web-GIS and hydrological model : a case study with google maps and IHACRES in the Oak Ridges Moraine area, Southern Ontario, Canada, In *Geoscience and Remote Sensing Symposium*, IGARSS, 4574–4577.

Bibliographie Web

Vérification datée du 01 mars 2010.

ANA : http://www2.ana.gov.br/Paginas/default.aspx
ArcGIS server : http://www.esrifrance.fr/ArcGIS_Server.asp
ATHYS : http://www.athys-soft.org/
Bibliographie de la géomatique sur le site GeoRezo : http://georezo.net/biblio.php
Bibliographie de référence de Ecole National des Sciences Géographiques (IGN) : http://www.ensg.eu/Bibliographies
Cemagref : http://www.cemagref.fr/
CNPq : http://www.cnpq.br/english/cnpq/index.htm
CNRS : http://www.cnrs.fr/
Comparaison entre les différents SIG de l'OSGeo : http://georezo.net/wiki/main:logiciels:comparatif_qgis_gvsig_udig_openjump#tableau_de_comparaison
Department of Geomatics, Melbourne University : http://www.geom.unimelb.edu.au/gisweb/
Dictionnaire de terminologie français – anglais du Québec : http://www.oqlf.gouv.qc.ca/ressources/gdt.html
DSI CNRS, gestion de projet : état de l'art des méthodologies : http://www.dsi.cnrs.fr/methodes/gestion-projet/methodologie/etat-art.htm
EPTB : http://www.debits-dordogne.fr/index.php?id=2
GeoServer : http://geoserver.org/display/GEOS/Welcome
Geotools : http://docs.codehaus.org/display/GEOTDOC/Home
Glossaire de l'école d'ingénieur de l'université de Melbourne : http://www.geom.unimelb.edu.au/gisweb/glossary.htm
Grass : http://grass.itc.it/index.php
gvSIG : http://www.gvsig.gva.es/(siteenlanguecatalane)
HYDROWIDE : http://www.hydrowide.com/
IRD : http://www.ird.fr/
IRD Brésil : http://www.brasil.ird.fr/spip.php?page=rubrique_accueil&id_rubrique=112
Java Native Interface (JNI) : http://java.sun.com/j2se/1.5.0/docs/guide/jni/spec/jniTOC.html
JNA : https://jna.dev.java.net/
JNative : http://sourceforge.net/projects/jnative/
Laboratoire Hydrosciences IRD/CNRS Montpellier : http://www.hydrosciences.org/
LISAH : http://www.umr-lisah.fr/
LMTG : http://www.lmtg.obs-mip.fr/
LTHE : http://ltheln21.hmg.inpg.fr/LTHE/index.php
Manifeste des méthodes agile : http://agilemanifesto.org/
Manuel de l'utilisateur OpenFLUID-Engine : http://www.umr-lisah.fr/openfluid/community/index.php/Documentation
MHYDAS : http://www.umr-lisah.fr/openfluid/index.php?page=mhydas
Mono : http://www.mono-project.com/

OGC : http://www.opengeospatial.org/
QGIS : http://www.qgis.org/
OMP : http://ezomp.omp.obs-mip.fr/
OpenFLUID : http://www.umr-lisah.fr/openfluid/
OpenJump : http://www.openjump.org/wiki/show/HomePage
OSGeo : http://www.osgeo.org/
OSGi : http://www.osgi.org/Main/HomePage
ProActive : http://proactive.inria.fr/
Refractions Research : http://www.refractions.net/
Standard UML par l'OMG ; http://www.omg.org/technology/documents/modeling_spec_catalog.htm
Sun Java : http://java.sun.com/
Terminologie géomatique (2005) : Office québécois de la langue française : http://www.oqlf.gouv.qc.ca/ressources/bibliotheque/dictionnaires/terminologie_geomatique/lex_geomatique.html
uDig : http://udig.refractions.net/
UPS : http://www.ups-tlse.fr/

Table des figures

Conception et développement d'Umodelis, une plateforme de modélisation hydrologique.

Mémoire d'ingénieur CNAM, Toulouse 02 juin 2010.

RÉSUMÉ

Umodelis est une plateforme de modélisation hydrologique distribuée. Son but est l'intégration des outils de traitements et données nécessaires à l'étude des bassins versants par les ingénieurs et les chercheurs de l'IRD Brésil et de leurs partenaires. L'architecture, qui repose sur J2EE, comprend un client Web, un client SIG facilitant la mise en forme des données des modèles et un serveur de modélisation chargé de la distribution des données, des modèles et de l'exécution des simulations. Le principe de modularité de la plateforme facilite l'intégration des outils et des concepts géomatiques existants et futurs. Umodelis propose, côté client, un module de création de projet à l'aide du SIG uDig et un module de contrôle de simulations et d'exploitation de leurs résultats. Côté serveur, elle implémente un module d'accès et de persistance des données et un module donnant un cadre aux modèles hydrologiques et permettant l'exécution des simulations.

Mots clés : modélisation, hydrologie, IRD, modulaire, J2EE, architecture distribuée, SIG.

SUMMARY

Umodelis is a distributed hydrological modeling platform. Its aim is to integrate processing tools and data necessary for engineers, scientists of the IRD Brazil and partners during watersheds studies. The architecture is based on J2EE and includes both a Web and a GIS clients. The latter facilitates the models' data formatting, whereas a modeling server is responsible for data and models distribution and run simulations. The modularity principle of the platform facilitates the integration of both existing and futur tools and geomatic concepts. On the client-side, Umodelis provides a project creation plugin using the uDig GIS, and a plugin of simulation control and operating results. On the server side, this platform implements a data access and persistence plugin, and a plugin providing framework for hydrological models which allows simulations runtime.

Keywords : modeling, hydrology, IRD, plugin, J2EE, distributed architecture, SIG.