Système d'**É**valuation et de **G**estion des **R**isques d'**I**nondation en milieu fluvial

Projet SEGRI

Rapport final N° R-720-F

Présenté au

Fonds des Priorités Gouvernementales en Science et en Technologie – volet Environnement (FPGST-E)

16 janvier 2006

Équipe de réalisation

Institut National de la Recherche Scientifique – Eau, Terre et Environnement

Yves Secretan	Professeur, PhD
Michel Leclerc	Professeur, PhD
Eric Larouche	Ingénieur en informatique
Paul Boudreau	Agent de recherche, M. Sc.-Eau
Nicolas Roy	Assistant de recherche

Collaborateurs

Maude Giasson	Stagiaire, étudiante à la maîtrise, boursière FCAR
Laurent Bonnifait	Étudiant à la maîtrise
Khalid Rahman	Étudiant à la maîtrise
Aurélien Mercier	Stagiaire (diplôme d'ingénieur), France
Sybil Christen	Stagiaire (diplôme d'études supérieure spécialisées)
Sébastien Quessy	Stagiaire (baccalauréat)
Olivier Bédard	Stagiaire (baccalauréat)
Thierry Malo	Stagiaire (baccalauréat), boursier CRSNG
Jean-Philippe Lemieux	Stagiaire (baccalauréat)
Sébastien Labbé	Stagiaire (baccalauréat)
Vincent Martineau	Stagiaire (baccalauréat)
Stéphane Lévesque	Stagiaire (baccalauréat)
Benjamin Behaghel	Stagiaire (diplôme d'ingénieur), France
Cédric Caron	Stagiaire (baccalauréat)
Daniel Nadeau	Stagiaire (baccalauréat), boursier CRSNG
Kevin Solinski	Stagiaire (diplôme d'ingénieur), France
Olivier Kaczor	Stagiaire (baccalauréat)
Renaud Le Boulleur de Courlon	Stagiaire (diplôme d'ingénieur), France
Samuel Ouellet	Stagiaire (baccalauréat), boursier CRSNG
Maxime Derenne	Stagiaire (baccalauréat), boursier CRSNG
Sébastien Bédard	Stagiaire (baccalauréat)
Dave Guérin	Stagiaire (technique)
Xavier Lavoie	Stagiaire (technique)
Dominic Richard	Stagiaire (technique)
Francis Larrivée	Stagiaire (technique)

Collaborateurs externes:

José Alfredo Bechara	Professeur	UNNE, Argentine
Maria de Lourdes Cavalcanti	Étudiante à la maîtrise	UFPA, Brésil
Manoel José dos Santos	Professeur	IESAM, Brésil
André Robitaille		Synexus Global
Michel Carreau		Synexus Global
André Plante		Environnement Canada
Jean Morin	PhD	Environnement Canada
Olivier Champoux		Environnement Canada
Daniel Rioux	M.Sc.	Environnement Canada
Jean Gauthier	M.Sc.	BPR
Pierre Bélanger		GPR
Richard Frenette	Professeur	Université d'Ottawa

Pour les fins de citation : **Secretan Y., Leclerc M., Larouche E., & coll. (2005).**

Système d'Évaluation et de Gestion des Risques d'Inondation en milieu fluvial (SEGRI) : Rapport final. Québec, INRS-Eau, Terre & Environnement. 128 pages. (INRS-Eau, Terre & Environnement, rapport de recherche R-720-F).

Pour: Fonds des Priorités Gouvernementales en Science et en Technologie – volet Environnement (FPGST-E).

ISBN : 2-89146-323-4

Table des matières

Table des figures

1 Introduction

Ce troisième rapport constitue le rapport final du projet SEGRI. Il couvre la période allant de janvier 2003 à décembre 2005 et conclut la dernière année du projet SEGRI.

L'objectif du rapport est de faire état de tout ce qui a été réalisé dans le cadre du projet, tant au niveau logiciel que scientifique au cours des trois années dernières années. Il vise également à présenter un bilan et à mentionner quelles devraient être les prochaines étapes.

Sur le plan scientifique, l'essentiel des efforts a porté sur l'analyse de risque dans ses différentes manifestations : coût des crues à l'eau libre (Leclerc et al., 2003; Blin et al, 2005) et des embâcles pour le secteur résidentiel (Leclerc et al, 2001, 2006), cartographie du taux d'endommagement ou risque unitaire (Blin et al., 2005), mise à jour des courbes du taux d'endommagement (Bonnifait et Leclerc, 2004, 2005).

Une première application a été entreprise dans la municipalité de Sainte-Marie de Beauce (de Courlon, 2003) et s'est poursuivie sur la rivière Châteauguay (Leclerc et al., 2006). Des travaux ont également été réalisés en rapport avec la mise à jour du plan de gestion des débits et niveaux d'eau du fleuve Saint-Laurent pour Environnement Canada et la Commission mixte internationale (CMI) (Doyon et al., 2004).

Sur le plan logiciel, Modeleur2 et H2D2, les logiciels au cœur du projet SEGRI, constituent les pièces tangibles des efforts de développement accomplis au cours du projet. Bien que certaines fonctionnalités restent à implanter et que l'aspect esthétique soit à retravailler, cette version contient la plupart des éléments clés du logiciel.

L'architecture élaborée au début du projet a été mise en place avec succès, et tel que prévu elle s'avère être souple et fonctionnelle. Le langage Python qui chapeaute plusieurs sections du logiciel est un choix qui s'est avéré judicieux, car il a permis de mettre en place aisément plusieurs aspects du logiciel (plug-in, interface usager, scripts). Le fait de développer autour d'une base de données (BD) relationnelle apporte beaucoup en ce qui a trait aux fonctionnalités pour les requêtes spatiales, en plus de valider intrinsèquement la cohérence entre les données. L'utilisation de librairies externes permet la réutilisation de code validé et d'ainsi concentrer les efforts de développement sur ce qui est propriétaire à Modeleur2.

L'un des défis majeurs au niveau logiciel était d'arriver à développer les logiciels avec une équipe de développement presque entièrement constituée de stagiaires. Bien que l'approche se soit avérée viable tout au long du projet, elle a aussi, jusqu'à un certain point, démontré ses limites. En effet, étant en quelque sorte dépendant des compétences des stagiaires recrutés, le flot de développement ne fut pas toujours aussi relevé qu'il aurait été requis.

Néanmoins, ce projet logiciel, par sa taille et sa complexité, s'est avéré être une plateforme d'apprentissage difficilement égalable. En plus des nombreux défis techniques à relever, les stagiaires se voyaient confier un niveau de responsabilités que peu d'organisations sont enclins à leur offrir. En contribuant de près à la formation de personnel hautement qualifié, le projet SEGRI a donc aidé à façonner le Québec de demain.

Le présent document est constitué de quatre sections. La section 2 effectue un rappel sur le contexte du projet SEGRI. La section 3 discute de l'aspect scientifique du projet. La section 4 présente les logiciels (Modeleur2 et H2D2), de même que les aspects environnement de développement et distribution des logiciels. Enfin, dans la section 5, il est question des ressources humaines et des tâches accomplies par chacune des personnes impliquées dans le projet.

2 Projet SEGRI

2.1 Problématique et contexte

Le projet vise la problématique des risques actuels d'inondation, ceux reliés aux éléments vulnérables existants et qu'on retrouve partout comme des faits accomplis sur les berges des cours d'eau. Nous croyons que les politiques actuelles interdisant les nouvelles constructions en zone inondables (Loi sur la qualité de l'Environnement, Loi sur l'Aménagement et l'Urbanisme) et auxquelles nous souscrivons entièrement sont arrivées trop tard dans bien des cas pour contenir le développement urbain en zones inondables. C'est un état de fait qui doit maintenant être adressé pour compléter l'approche préventive des politiques publiques actuelles.

De plus, les risques reliés aux embâcles forment la majeure partie des dommages indemnisés dans un pays froid comme le nôtre. Les ignorer serait envoyer un mauvais signal aux aménagistes en autorisant des constructions dans des endroits aussi sinon plus menaçants que les crues à l'eau libre. L'estimation de ce risque pose cependant le défi de la reconstitution de séries fiables relatives à ces événements.

Nous espérons donc, par une approche curative complémentaire prenant acte de la situation présente, toutes classes d'aléas confondues, contribuer à mieux diagnostiquer la nature et le coût récurrent des risques actuels à l'échelle du tronçon hydrographique, et à les résoudre. Puisqu'elle contribuera à réduire à long terme les dommages, elle a également les qualités d'une approche préventive.

2.2 Buts et objectifs

Le projet vise à développer une méthodologie intégrée qui cherche d'abord :

- à établir le bilan des risques actuels d'inondations sur un cours d'eau ;
- à élaborer une stratégie intégrée de réduction des risques qui se justifie par sa rentabilité économique, tout en respectant des critères de durabilité, de respect du milieu naturel et d'équité entre les divers intervenants à risque du bassin ;
- à proposer une gestion des risques résiduels par une concertation des intervenants.

Il vise aussi à démontrer la faisabilité et la pertinence de l'approche par une application au tronçon de la rivière Châteauguay à Châteauguay[1]. Des outils et méthodologies validés seront ainsi élaborés en vue de leur application sur d'autres bassins, pour d'autres MRC et municipalités.

[1] Le plan original prévoyait une application à Sainte-Marie de Beauce mais en raison des coûts importants qu'une telle application entraîne, c'est finalement à Châteauguay que l'application a pu être réalisée avec le support financier de la Ville et du ministère de la Sécurité publique. Par contre, l'analyse des conditions de submersion en crue à l'eau libre a pu être réalisée par de Courlon, 2002.

Nos travaux précédents ont mis en lumière divers goulots d'étranglement qui doivent être adressés pour rendre les outils génériques et utilisables pour les applications visées. Nous avons besoin d'un modèle de terrain précis pour calculer les profondeurs de submersion, et d'une modélisation fine pour évaluer l'impact de changements morphologiques. Nous devons évaluer localement les dommages et les risques.

Puisque la précision des réponses fournies dépend en grande partie de la précision des mesures de terrain, la qualité et la quantité de ces mesures deviennent critiques pour la rentabilité et l'accessibilité des projets de modélisation.

2.3 Méthodologie

Une méthodologie où les principes d'intervention sont formulés explicitement (équité, non-transférabilité, durabilité) et de nouveaux instruments de calcul informatisés faisant appel à la géomatique et à modélisation numérique seront élaborés. Les variables visées sont l'altitude du terrain en zone inondable (géomatique), les débits naturels ou influencés (hydrologie), les niveaux d'eau et les vitesses de courant (hydraulique), le potentiel de vulnérabilité en zone inondable (S.I.G.) ainsi que les coûts des dommages (économétrie).

Le projet se veut donc axé sur le développement d'instruments précis et innovateurs qui seront mis à l'épreuve dans un contexte de démonstration sur une rivière.

2.4 Résultats escomptés

Les résultats escomptés sont de deux sortes :

- une méthodologie d'analyse et de réduction des risques d'inondation appliquée et validée sur une rivière ;
- un logiciel qui soutient cette méthodologie.

La méthodologie s'appuie sur les cartes de risques qui mettent en évidence des zones sous-protégées et des zones surprotégées, synthétisent la situation existante mais aussi les aménagements projetés. Ces cartes sont la base d'une compréhension du risque d'inondation.

Le logiciel est dans la lignée des MODELEUR/HYDROSIM et HABIOSIM. Il en réutilise la base de donnée et les fonctionnalités de base. C'est un outil de traitement spatial spécialisé, assimilable à un Système d'Information Géographique (S.I.G.).

2.5 Progrès environnemental

Le présent projet est innovateur par l'accent mis sur l'intégration hydro-environnementale de modèles de terrain détaillés, de modèles hydrauliques et de modèles d'évaluation des dommages, et par la convivialité du système. Il est innovateur par ses méthodes numériques performantes, sa programmation orientée-objet, ses fonctionnalités informatiques et graphiques, son caractère multidisciplinaire et son approche de partenariat. C'est probablement la première fois que l'on mettra à la disposition des intervenants du milieu hydrique un outil permettant de faire des études détaillées de la dynamique des inondations et de la vulnérabilité des zones riveraines. Pour ces raisons, il se situe d'emblée à l'avant-garde de telles démarches au Canada, voire, dans le monde.

L'intérêt se situe globalement à trois niveaux :

- minimiser les risques humains et matériels reliés aux inondations ainsi que leurs coûts récurrents ;
- développer une technologie générique, économique, durable et environnementalement viable pour contrôler les risques d'inondation ;
- doter les municipalités d'une habilité accrue à gérer le risque résiduel basée sur une meilleure connaissance du risque sur les territoires municipaux.

Le système proposé adresse des domaines d'intervention dont les coûts pour le Québec et le Canada peuvent atteindre des millions de dollars récurrents. Une diminution des risques serait de nature à diminuer les coûts ou à accroître l'efficacité des mesures.

2.6 Potentiel commercial

Pour avoir conduit une étude de marché pour les logiciels MODELEUR/HYDROSIM dans le cadre du projet MÉTRIQUE, il est apparu qu'il existe un marché pour des logiciels d'hydraulique fluviale, mais qu'il n'est pas très grand. Par contre, comme la prévention des risques d'inondations est une responsabilité ultime des gouvernements fédéral, provincial et municipal, l'intérêt public est manifeste. Dans ce sens, il s'agit beaucoup plus d'un marché de services de consultation que l'on peut rendre avec l'outil, que d'un marché pour l'outil. L'engagement de trois partenaires privés dans le projet le montre bien.

En permettant de prouver la rentabilité économique d'interventions pour réduire les risques liés à l'occupation actuelle des zones inondables, le projet va ouvrir le marché de la cartographie des risques aux consultants ainsi qu'aux entreprises de géomatique, et plus particulièrement au laser aéroporté. À cet égard, il renforce la place grandissante du Québec sur le marché de la géomatique. Il pave également la voie à une politique d'assurabilité des zones d'inondation en permettant de déterminer avec précision les risques encourus et la rentabilité des actions.

La cartographie des risques d'inondation intéresse directement les municipalités qui doivent en tenir compte dans leurs plans d'aménagement, les compagnies d'assurance qui seraient en mesure d'offrir des produits d'assurance dans ces zones, mais aussi la protection civile qui aurait une base pour planifier des interventions en cas d'inondation. Il est essentiel pour les firmes canadiennes et québécoises de pouvoir offrir une technologie informatique originale leur permettant de se démarquer de l'offre internationale. C'est dans l'exclusivité de ses outils et dans son savoir-faire original dans un marché de services que se trouve l'intérêt économique du Québec.

3 Les risques d'inondation

En zone à risque, les dommages d'inondation peuvent s'expliquer par la vulnérabilité des bâtiments à la submersion. Cette variable prend la forme d'un champ distribué (x,y) pour chaque événement considéré. L'aspect spatialisé de la submersion fait de cette variable un objet de prédilection pour un outil géomatique comme MODELEUR. D'autant plus que l'outil est dédié à l'hydraulique fluviale et qu'il permet de conduire des simulations très précises des écoulements dans toutes les gammes de débits, et ce en mode prédictif avec H2D2, son module hydrodynamique couvrant-découvrant. De multiples fonctions de représentation des données spatiales et la disponibilité d'une calculatrice programmable permettent de réaliser des calculs complexes comme les dommages de submersion avec une précision inégalée, équivalente au minimum à celle obtenue au niveau topographique.

C'est dans cette optique qu'ont été mis en œuvre successivement deux projets d'analyse de risque d'inondation sur la rivière Chaudière et la rivière Châteauguay.

3.1 Sainte-Marie de Beauce

L'étude du premier cas (Chaudière à Sainte-Marie de Beauce) a été entreprise au tout début du projet SEGRI, c'est-à-dire, avant que les nouvelles fonctionnalités de la version 2 de MODELEUR soient mises en œuvre. Les premiers travaux ont donc été initiés avec la version 1 de l'outil. Plusieurs réunions de travail ont été organisées à Sainte-Marie avec les autorités municipales et les partenaires du projet, notamment le CEHQ.

Les travaux réalisés par de Courlon (2003) ont permis de constituer un modèle numérique d'élévation (MNE) complet des quartiers bas de Sainte-Marie de Beauce, de caractériser la topographie du lit mineur, et d'implanter le simulateur hydrodynamique HYDROSIM sur l'ensemble du secteur d'étude. Les simulations ont permis de mieux comprendre la logique des écoulements dans ce secteur à risque urbanisé et industrialisé. Cette analyse a permis d'identifier les nombreux empiètements du lit mineur, notamment en rive du côté droit et en travers de la section d'écoulement à la hauteur du pont de Sainte-Marie (ses approches), comme une des causes principales des surcotes observables à l'amont du pont et ayant pour effet de submerger les quartiers industriels de la ville.

Au moment où le ministère des Transports du Québec planifiait la construction d'un nouveau pont pour remplacer l'ancien devenu désuet, des démarches ont été entreprises par la Ville afin de concevoir et dimensionner un ouvrage, et recalibrer les rives à son voisinage, pour faire en sorte d'atténuer les surcotes provoquées par la restriction existante. Parmi les solutions envisagées, on comptait le rétablissement d'une section suffisante pour transiter les crues à une cote inférieure à la situation actuelle, le moyen étant l'élargissement du lit mineur par un déblai en rive gauche à la hauteur du pont. L'approche du pont en rive gauche constituant également un empiètement majeur de la zone inondable, il était également envisagé de transformer cette section de voirie en zone ouverte à l'écoulement par la mise en place d'une infrastructure de ponceaux sous-jacents.

Les démarches entreprises par la Ville n'ont pas reçu l'accueil souhaité auprès du ministère qui a réalisé sa planification de manière traditionnelle, c'est-à-dire, sectorielle, sans tenir compte des possibilités de bénéfices accrus qu'aurait procurés une solution intégrée, notamment par la réduction du risque.

Étant donné le manque d'intérêt manifesté par l'organisme (MTQ) qui aurait eu le pouvoir et l'opportunité d'apporter des solutions à la problématique pourtant lourde des risques d'inondation à Sainte-Marie, l'équipe du projet SEGRI s'est détournée de ce cas pour jeter son dévolu sur une autre situation aussi lourdement problématique, le risque d'inondation à Châteauguay. Il faut aussi ajouter que le coût de réalisation d'une étude complète des inondations, allant jusqu'à formuler des recommandations précises et chiffrées en terme de risque, s'avérait trop élevé pour les ressources disponibles du projet SEGRI et que la Ville ne contribuait pas financièrement à sa réalisation (sauf contributions en nature comme la topographie terrestre).

3.2 Châteauguay

À l'automne de 2004, l'INRS-ETE a été sollicité pour participer à la recherche d'une solution intégrée à la problématique des inondations à Châteauguay. Cette invitation s'accompagnait de ressources financières et en nature importantes, bien que limitées, et par l'implication directes de nombreux partenaires intéressés par une telle étude, notamment, la MRC, le CEHQ, le MDDEP, le Conseil de bassin, la Ville de Châteauguay, la Communauté métropolitaine, Environnement canada, le Consortium Ouranos, les firmes de consultants et surtout le ministère de la Sécurité publique qui est très préoccupé par les coûts exorbitants des inondations à Châteauguay. Nous y avons vu là une occasion exceptionnelle de mettre en œuvre les idées émergeant du projet SEGRI, d'en évaluer la faisabilité et de munir les logiciels développés des fonctionnalités requises pour adresser certaines tâches plus efficacement.

L'approche de l'analyse de risque n'est pas nouvelle et on en trouve plusieurs exemples dans la bibliographie du MEDDP datant des années 80'. Le caractère plus rudimentaire des outils utilisés alors et la portée limitée donnée alors au risque, lequel ignorait par exemple, la part de dommages assumés par les riverains, ou encore, les dommages d'incertitude qui se traduisent en réduction de l'assiette fiscale foncière en zone à risque, limitait alors les possibilités d'intervention remédiatrices. L'extension de la portée du risque à ces nouveaux éléments crée de nouvelles perspectives d'intervention dont on voit maintenant le potentiel de rentabilité accru.

La Figure 1 montre un schéma de la méthodologie développée dans le cadre de SEGRI et appliquée intégralement à Châteauguay. Si l'approche décrite semble naturelle et relativement classique, son application intégrale compte très peu sinon pas de cas similaire dans les études antérieures du moins à l'échelle du Québec. L'étude de Châteauguay (Leclerc *et al.*, 2006) a poussé très loin toutes les dimensions innovatrices de l'approche.

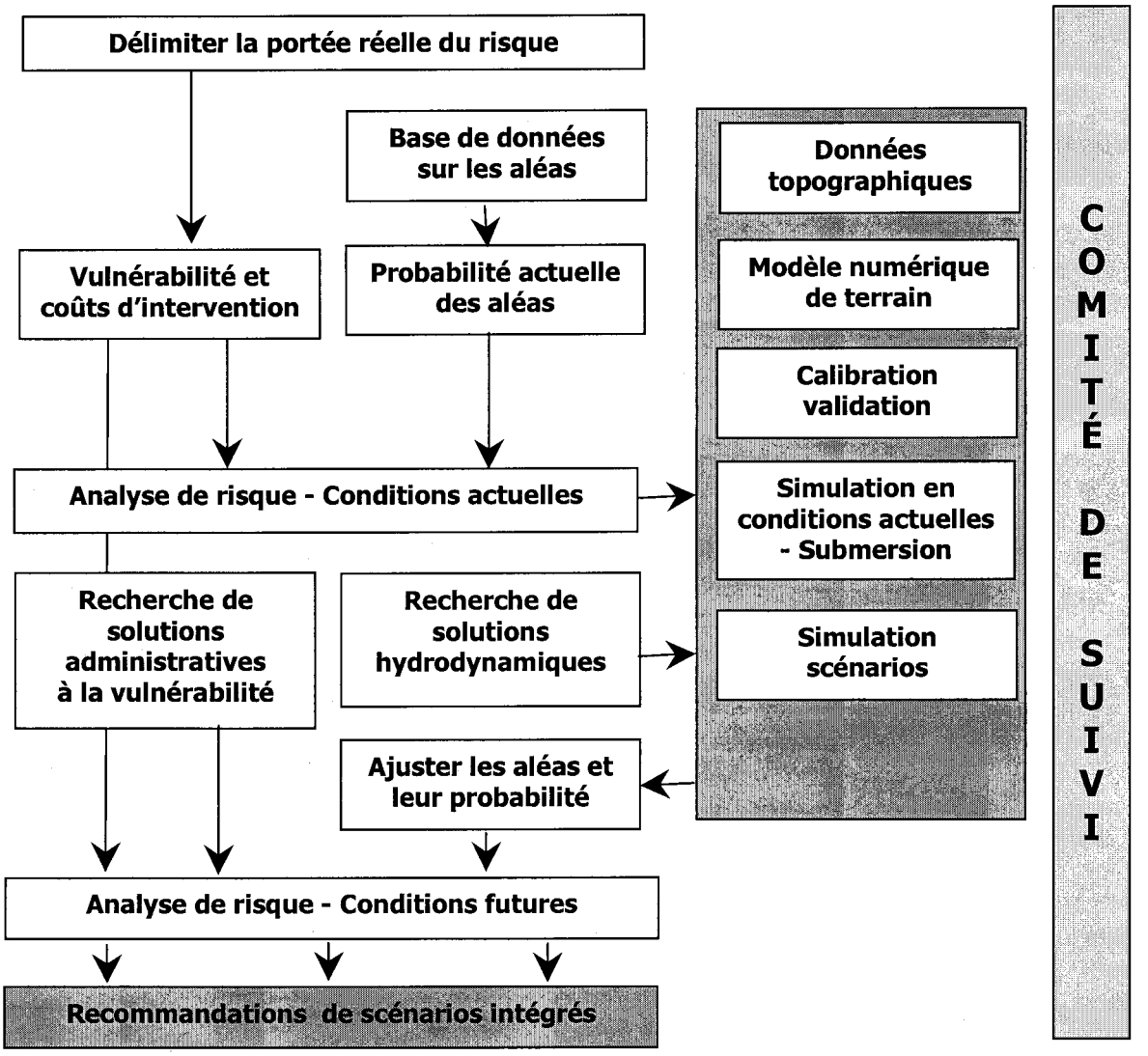


Figure 1 - Méthodologie d'analyse du risque d'inondation

La méthode permet de poursuivre les objectifs spécifiques suivants :

1. *D'étendre la portée à considérer des enjeux économiques représentés par les coûts et l'équité du risque, incluant les dommages directs, le coût des interventions ainsi que les dommages d'incertitude*
2. *De regrouper les données disponibles sur les événements problématiques sous la forme d'une base de données chronologiques axée autant que possible sur le niveau d'eau maximum atteint*
3. *De mettre à jour la probabilité d'occurrence des inondations incluant des hypothèses sur les changements climatiques*
4. *De déterminer les conditions de submersion des crues à l'eau libre et des embâcles*

5. *De regrouper les données disponibles sur la vulnérabilité des constructions : hauteur et nombre de planchers, présence et état de finition du sous-sol, valeur foncière, exposition à la submersion*
6. *D'établir la valeur et la distribution spatiale du risque*
7. *Identifier les solutions potentielles, de nature administrative ou structurale, et en quantifier dans la mesure du possible l'efficacité*
8. *Proposer une solution intégrée*

La méthodologie est détaillée ci-après. Ce passage est inspiré d'un chapitre du rapport de Leclerc *et al.* (2006). Il est repris ici dans ses aspects génériques. Il est à noter que son application intégrale en démontre la faisabilité malgré certaines difficultés résiduelles, notamment celles reliées à la compréhension physique et statistique du phénomène des embâcles, pour lequel beaucoup de travail reste à accomplir.

3.2.1 Tenir compte de la portée réelle des coûts du risque

Dans la plupart des études de risque, les dommages considérés se limitent souvent aux coûts directs d'intervention pris en charge par l'intervenant principal au dossier, habituellement le gouvernement, qui peut ainsi évaluer pour lui-même l'opportunité d'intervenir en remédiation d'après le rapport coûts/bénéfices attendu. Cette limitation tend à produire une image tronquée des coûts du risque. La vraie portée des dommages d'inondation s'étend aux autres intervenants (riverains, municipalités), chacun d'eux assumant sa part d'équité des divers dommages ou coûts.

Ainsi, il est généralement reconnu (voir les décrets d'indemnisation) qu'au moins 50% des dommages directs aux résidences (notamment les pertes de bien non essentiels) doivent être assumés par les sinistrés, en plus des stress psychologiques et inconvénients subis. Les interventions directes sur la rivière sont en partie assumées par la Ville à des degrés divers selon que le contexte est préventif, urgent (gestion de crise) ou post-événementiel.

Plus insidieusement, on observe une perte importante de la valeur marchande (patrimoniale) des propriétés notoirement à risque (*dommages d'incertitude*) qui se traduit par une valeur au rôle minorée, donc à un manque à gagner fiscal significatif pour les pouvoirs locaux (taxes municipales et scolaires). Cette situation a été observée sur la rivière Montmorency qui est aussi notoirement à risque d'embâcles (Île-Enchanteresse; voir Leclerc *et al.,* 2001). Ce manque à gagner se traduit par un compte de taxe inférieur à ce qu'il devrait être pour les sinistrés récurrents, facteur qui, ajouté à une valeur marchande dépréciée, peut insidieusement devenir incitatif pour le maintien de ces populations en zone inondable. Cette problématique existe aussi dans la ville de Châteauguay et présumément dans toute région soumise à des risques notoires.

Si les coûts des dommages directs peuvent être estimés à partir des données d'indemnisation antérieures, cette approche dite « historique » permet difficilement

d'effectuer des prédictions reliées à des scénarios d'intervention. Comme exposé plus loin, nous proposons une méthodologie de prédiction des dommages directs plutôt basée sur la géomatique qui fait intervenir les paramètres de vulnérabilité des bâtiments et leur exposition à la submersion.

Concernant les dommages d'incertitude, ils peuvent être estimés en posant des hypothèses raisonnables sur la valeur actuelle dépréciée des bâtiments et celle qu'elles auraient suite à l'application de divers scénarios remédiateurs. L'appréciation foncière résultant de la relocalisation ou de la réduction, voire l'élimination des incertitudes reliées aux aléas doit normalement se traduire à plus long terme par une appréciation de l'assiette fiscale des pouvoirs locaux et la restauration du patrimoine des individus.

***En résumé**, la portée du risque est étendue aux dommages d'incertitude sous la forme de leurs implications fiscales, et une méthode géomatique d'estimation des dommages directs est utilisée pour la prédiction du résultat des interventions.*

3.2.2 Construire une base de données sur les aléas de crue et d'embâcles

Le calcul du risque nécessite de connaître la probabilité des crues et embâcles qui sert de pondération pour la valeur des dommages. Il est relativement aisé d'établir la fréquence théorique (dite marginale) des débits de crues grâce à la disponibilité de séries chronologiques homogènes sur la plupart des cours d'eau importants et à l'état de l'art très avancé des statistiques de crues à l'eau libre. Cependant, en certaines circonstances comme à la confluence de deux cours d'eau, comme à Châteauguay, les paramètres statistiques de la série disponible (station 030905) doivent être revisités pour tenir compte de l'effet conjugué du niveau du cours d'eau majeur (fleuve Saint-Laurent) sur l'affluent, facteur qui peut à lui seul devenir cause d'inondation, du moins à l'embouchure du cours d'eau.

Il est donc requis d'analyser les données de débit disponibles sur l'affluent conjointement avec les niveaux du cours d'eau majeur. Une approche totalement nouvelle de cette problématique en mettant l'accent sur les probabilités conjointes (méthode des copules) est proposée pour analyser ces situations particulières. Dans le cas des petits bassins versants (typiquement inférieurs à 2000-5000 km^2), les valeurs maximums horaires doivent être considérées plutôt que les valeurs moyennes journalières en raison du caractère soudain de certaines de leurs crues (éclair) associé à leur temps de réponse court, inférieur à la journée.

Le niveau d'eau des rivières est une donnée sujette à évoluer à plus ou moins long terme, notamment en raison des changements climatiques et à la modification de plans de gestion de ses débits et niveaux d'eau (ex : le système Saint-Laurent/Grands-Lacs géré par la CMI). En vue de prendre acte de l'évolution future du plan de gestion et des tendances des changements climatiques, des hypothèses doivent être posées afin de mesurer l'impact de ces influences sur la valeur du risque.

Concernant les embâcles, l'absence de séries chronologiques comparables aux débits oblige d'abord travailler à la reconstitution des niveaux d'eau et des conditions d'occurrence pour les événements connus. La donnée peut être parfois être obtenue (déduite, estimée) à partir des rapports existants et d'indices disponibles comme les documents photographiques, les témoignages, ou les carnets de terrain. Des données indirectes concernant les indemnisations accordées, voire les demandes d'indemnisation peuvent également constituer une indication de niveau atteint. L'usage de la dendrochronologie et l'observation des cicatrices sur les arbres laissées par la glace et les débris peut également apporter une contribution à cette connaissance.

***En résumé**, une base de données des aléas de crues et d'embâcles est constituée en mettant l'accent sur la série des maximums annuels horaires de débit si nécessaire, sur les facteurs conjoints (ex : niveaux maximums du fleuve),et sur l'ensemble des données événementielles d'embâcles disponibles chez les divers intervenants.*

3.2.3 Établir la probabilité actuelle et évolutive des aléas

Une méthodologie statistique réaliste visant à établir la probabilité individuelle ou conjointe des aléas est appliquée à la base de données évoquée au paragraphe précédent. Les méthodes disponibles y ont été évoquées mais encore faut-il les adapter au contexte particulier à l'étude, notamment à l'égard des embâcles et de la conjugaison des éléments générateurs des aléas. Par exemple, la combinaison de facteurs est très significative dans le secteur de Châteauguay pour les éléments suivants: crues de la rivière, embâcles à l'embouchure, crues du fleuve, plan de gestion des débits et niveaux d'eau en cours de révision par la CMI, changements climatiques. Dans la mesure du possible, une convergence méthodologique vers les niveaux d'eau est recherchée dans le but de traduire cette information en hauteur de submersion.

Concernant les embâcles, le faible échantillon tiré des indications disponibles sur le niveau d'eau ne permet pas la plupart du temps de modéliser sa structure statistique et oblige l'utilisation de probabilités empiriques. D'ailleurs, il n'existe pas à notre connaissance de méthode statistique rigoureuse visant à établir la probabilité théorique de tels événements. L'absence de séries chronologiques continues (comme les débits) et à long terme, encore moins d'une base de données formelles comme le suggère White (1996) du CRREL (US Army Corps of Engineers) et l'hétérogénéité des informations disponibles sur les embâcles connus ne facilitent pas leur analyse statistique.

Une méthode de détermination des probabilités empiriques des risques d'embâcles a été proposée par Leclerc *et al.* (2001; 2005) sur la Montmorency. Elle est basée sur une classification des événements à partir d'indices hétérogènes par nature (témoignages, cicatrice sur les arbres, indemnisations, etc.), dont parfois la mesure du niveau d'eau maximum atteint. Cependant, l'approche citée n'est pas prédictive pour des projets d'intervention sur le cours d'eau, puisqu'elle repose sur des données historiques.

Comme les interventions structurales qui peuvent résulter d'une telle étude auront des effets quantifiables sur la submersion, la structure de probabilité des aléas doit donc être

revisitée pour chaque scénario afin d'établir la rentabilité des solutions envisagées (rapport coûts/bénéfices). Dans ce cas, des hypothèses raisonnables sont posées pour anticiper le changement, ce qui laisse place à une marge d'incertitude plus importante que pour les conditions actuelles. Il s'agit alors de demeurer conservateur (sécuritaire) vis-à-vis des estimations réalisées.

Concernant les changements climatiques, seules des hypothèses probabilistes non validables à court terme, et basées sur les tendances du climat et ses impacts sur les phénomènes d'inondation, peuvent être posées à titre indicatif.

***En résumé**, une sélection des méthodes les plus appropriées est effectuée pour déterminer la probabilité théorique et/ou empirique, individuelle ou conjointe, des différentes classes d'aléas causant les inondations, dans les conditions actuelles, en présence d'aménagements remédiateurs ou en fonction des changements climatiques anticipés.*

3.2.4 Simuler les conditions hydrauliques d'inondation

Cette activité, la plus classique dans le cadre de SEGRI, vise à mieux comprendre la distribution des écoulements dans le tronçon de référence. Compte tenu de la complexité relative de la topographie du tronçon, l'utilisation de modèles bidimensionnels (2D) peut être indiquée, du moins pour les conditions à l'eau libre. Pour les embâcles, une modélisation 1D à l'aide du module d'embâcle de HEC-RAS (ou tout autre outil équivalent) est retenue. Le projet SEGRI ne comprenait pas d'activité de développement spécifique pour ce type de phénomène, sauf pour assurer une meilleure communication entre MODELEUR et le simulateur mentionné, par exemple :

1. assimiler des transects HEC dans le modèle numérique d'élévation (MNE) 2D,
2. ajouter des transects 1D dans HEC à partir du MNE 2D
3. transposer des résultats 1D de HEC dans l'espace 2D de la zone inondable,
4. délimiter des zones inondables de période de retour donnée, etc.

L'activité de modélisation comprend les étapes classiques suivantes :

1. Collecte des données de terrain existantes : sections HEC, données laser aéroportées d'Environnement Canada, caractérisation topographique complémentaire du lit mineur (bathymétrie
2. Construction du modèle numérique de terrain 2D et 1D dans un SIG approprié (MODELEUR V1)
3. Choix d'événements de référence pour la calibration-validation des modèles hydrodynamique et d'embâcles et implantation de ces modèles
4. Simulation d'une gamme d'intérêt de conditions de crue et d'embâcle (sensibilité aux facteurs déterminants comme conditions aux limites)
5. Identification, insertion, paramétrisation et simulation des aménagements remédiateurs (voir plus loin)

6. Calcul de la submersion et estimation des dommages directs

Bien que la modélisation soit une étape majeure de l'étude, elle n'est évoquée que brièvement ici, le lecteur étant référé aux publications appropriées. Pour les crues à l'eau libre, MODELEUR/HYDROSIM est utilisé.

***En résumé**, la simulation numérique 1D et 2D des conditions d'écoulement actuelles et des divers scénarios remédiateurs est retenue. Le but de l'activité est de calculer la submersion due aux inondations pour chaque scénario étudié, et subséquemment les dommages directs.*

3.2.5 Regrouper les données sur la vulnérabilité des constructions

Cette activité vise à modéliser les dommages directs en fonction de la submersion pour chaque événement de référence, qu'il soit actuel ou potentiel. La méthodologie est donc géomatique et requiert la mise en place d'une base de données immobilières spatialisée, incluant la position des bâtiments dans le terrain (leur exposition aux aléas, l'altitude des planchers), le type de construction (présence de sous-sol, immunisation) et leur évaluation foncière qui sont les principaux déterminants des dommages directs. La plupart de ces données sont disponibles sauf la cote d'implantation des bâtiments laquelle peut nécessiter une caractérisation.

L'approche développée pour les dommages directs se limite aux immeubles résidentiels qui constituent, sauf exception comme à Sainte-Marie de Beauce la grande majorité des bâtiments vulnérables aux inondations en milieu urbain. L'étude de Châteauguay a permis de mobiliser une base de données comportant plus de 2000 entités résidentielles soumises à un degré de risque non nul. Elle a aussi permis de mettre au point une méthode de caractérisation visuelle des hauteurs de plancher (rez-de-chaussée) très efficace.

***En résumé**, l'activité vise la mise en place d'une base de données résidentielle avec comme variables les déterminants de la vulnérabilité aux dommages.*

3.2.6 Établir la distribution des dommages et du risque

Cette activité produit l'essentiel des résultats recherchés, c'est-à-dire, la distribution spatiale des dommages par événement ciblé, le dommage total par événement ainsi que la valeur cumulative du risque tous événements confondus. L'approche consiste à calculer la hauteur de submersion à la position géoréférencée des bâtiments résidentiels et d'appliquer une relation fonctionnelle convertissant cette submersion en dommages directs. Nous proposons d'utiliser et adapter les courbes de Bonnifait (2005) développées en collaboration avec Environnement Canada (Doyon *et al.*, 2004; Bonnifait et Leclerc, 2004) pour le compte de la Commission mixte internationale (CMI). Lesdites courbes ont été développées à partir de données du Saguenay (crue de 1996) du lac Saint-Louis (dont une partie provient de Châteauguay, crues de 1996 et de 1998) et de la région des îles de Sorel (crue de 1998) ce qui leur confère un caractère assez générique pour l'habitat québécois. Il est également possible en appliquant la méthode géomatique distribuée de

Blin (2002) et Blin *et al.* (2005) de cartographier le risque lui-même en terme de taux d'endommagement toutes classes d'événements confondues pour certains types de construction.

La question des refoulements des réseaux de drainage urbain est une dimension particulière de la vulnérabilité à la submersion des résidences munies de sous-sols qui peut difficilement être prise en compte sans des données précises sur l'état de ces infrastructures. Cette problématique se distingue de la submersion directe à partir du terrain qui fait l'objet de ce rapport. La méthodologie de calcul de risque proposée ne prend pas en compte ce phénomène bien que son importance soit indéniable dans plusieurs municipalités (ex : la crue de 1998 à Châteauguay, la rivière Lorette à l'automne 2005). La submersion des sous-sols liée à l'inondation du terrain n'est pas beaucoup plus simple à traiter car elle dépend de la présence d'ouvertures ou fissures dans les fondations, une donnée individuelle trop particulière pratiquement impossible à incorporer dans un modèle de dommages. Des hypothèses sont requises pour en tenir compte de manière simplifiée.

Innovation importante de l'approche proposée, la définition du risque est étendue aux dommages d'incertitude. Pour ce faire, des hypothèses raisonnables sur le facteur « incertitude » et ses conséquences sur la valeur foncière des bâtiments doivent être posées afin d'induire des conclusions sur l'assiette fiscale et son rendement. D'évidence, l'incertitude peut dépendre de la fréquence des événements dépassant une certaine magnitude; elle peut aussi décroître en fonction du temps, du laps de temps depuis le dernier sinistre, et surtout du succès des interventions. La détermination des paramètres de calcul peut être obtenue par avis d'experts (ex. : évaluateurs) et par analyse différentielle des valeurs existantes dans la ville.

***<u>En résumé</u>**, une méthode géoréférencée d'estimation des dommages directs basée sur la submersion simulée et sur la vulnérabilité des bâtiments existants est appliquée. Pondérés par leur fréquence événementielle relative, les résultats constituent le risque de dommage direct (moyenne annuelle). Le dommage d'incertitude est estimé selon la perte de valeur à long terme et sa répercussion sur l'assiette fiscale municipale et scolaire. La question des refoulements n'est pas traitée bien qu'elle puisse constituer une part significative des dommages directs.*

3.2.7 Rechercher des solutions et évaluer leur efficacité technique

Les solutions préventives comprennent d'abord les moyens déployés en gestion de crise (ex : alerte précoce, évacuations), lesquels visent surtout les pertes de vies et les inconvénients importants reliés aux inondations. Les moyens préventifs sont aussi de type structural (exemple : estacades, dérivations) et/ou administratif (ex : relocalisations, immunisations).

Les mesures structurales visent par des aménagements particuliers du lit du cours d'eau ou des ouvrages de rétention ou de dérivation de l'eau ou de la glace, à atténuer l'aléa ou à le déplacer en des sites où aucune vulnérabilité (immeubles exposés) n'est présente. Si

une certaine vulnérabilité existe en ces endroits, des relocalisations peuvent être requises ce qui n'est pas souhaitable bien entendu, à cause de la résistance sociale et des coûts additionnels engendrés. Il existe toute une panoplie de solutions structurales qui devront être considérées. À part certaines plus radicales (ex : barrage Sartigan sur la Chaudière, notamment), il n'est pas toujours possible de garantir le degré d'efficacité de telles mesures si bien que leur dimensionnement relève autant de l'art que de calculs précis. Des hypothèses raisonnables d'efficacité doivent donc être posées pour en estimer la rentabilité.

Des études antérieures ont déjà permis d'identifier plusieurs types de solutions structurales, notamment Tuthill (1995) du Cold Region Research and Engineering Laboratory (CRREL, US Army) et Morse *et al.* (2002). Parmi ces dernières, notons les estacades à peigne ou flottantes. Une étude de remédiation doit identifier et analyser un sous-ensemble de mesures possibles dans le secteur étudié dans une optique de pré-faisabilité. Le type, la localisation et le gabarit caractéristique des mesures structurales sont déterminés et leur efficacité est paramétrisée dans la mesure du possible en terme de réduction du niveau d'eau dans la zone d'influence pour les événements ciblés, donc en terme de réduction des dommages et du risque.

Les moyens administratifs sont d'abord constitués de règles d'interdiction ou de restrictions (ex : immunisation obligatoire) à la construction ou à la rénovation en zone inondable, le but étant de stopper la progression de la valeur du capital physique en zone vulnérable qui s'accompagne automatiquement d'une croissance du risque. S'il ne permet pas une réduction du risque, ce moyen permet du moins d'en stopper l'augmentation. Cette approche s'exprime sous la forme de cartes (exemple : Convention Canada/Québec) dont la confection ne relève pas de notre mandat.

Un autre moyen administratif plus radical consiste à relocaliser les résidences ou bâtiments à risques dans des zones non vulnérables. Dans ce cas, la réduction du capital physique soumis aux aléas est diminuée d'autant, ce qui réduit *ipso facto* le risque. De plus, si la résidence est relocalisée dans la même ville (souhaitable), la plus-value possible apportée à la propriété se traduit en avantages fiscaux pour la ville et en gain patrimonial pour le riverain. Toutes ces mesures peuvent rencontrer des résistances de la part des propriétaires riverains si le plan de mise en œuvre n'est pas suffisamment équitable par rapport à ceux-ci (principe « win-win »).

Dans le cas des relocalisations (ou immunisations complètes), la méthode de calcul consiste à retirer du calcul de risque les propriétés vulnérables dans un ordre de priorité défini par la valeur croissante du risque encouru, donc du degré d'exposition à la submersion. C'est donc la réponse à la submersion qui devient la variable charnière pour tous les cas de figure d'intervention.

Dans un calcul de rentabilité, les investissements sont aussi estimés et mis dans la balance sur la base d'un coût moyen annuel à comparer au gain anticipé. Des avantages indirects (ex : meilleure équité fiscale municipale, patrimoines individuels majorés) reliés à la relocalisation font aussi partie des avantages et doivent être comptabilisés à l'actif. Une

telle approche présente l'avantage de prioriser les interventions à entreprendre même dans l'optique où d'autres considérations que la rentabilité économique (ex : assurer la sécurité des riverains, changer la vocation des couloirs riverains dans une optique d'accessibilité) militent en faveur des interventions.

***En résumé**, une caractérisation sommaire des différents types de solutions remédiatrices et leur analyse d'efficacité pour le contrôle du risque sont réalisées. L'accent est mis soit sur la réduction du niveau d'eau, ou sur la gestion de la vulnérabilité du parc résidentiel. L'analyse relative aux moyens administratifs vise surtout à identifier un rationnel permettant de désigner les propriétés qui pourraient en priorité être considérée pour une relocalisation. Une approche économique axée sur le rapport coûts/bénéfices est appliquée lorsque c'est possible.*

3.2.8 Proposer une stratégie intégrée

À la lumière des résultats anticipés de chacune des mesures envisagées, une sélection est effectuée afin de minimiser le risque. Dans les limites du critère de rentabilité économique, il n'est sans doute pas réaliste d'anticiper l'élimination complète du risque. Par contre, certaines interventions s'imposent automatiquement. Au-delà de la rentabilité globale des interventions, toutes catégories de dommages et coûts confondues, il faut aussi rechercher l'équité entre les intervenants, c'est-à-dire que le rapport coûts/bénéfices doit être avantageux pour toutes les parties prenantes (le principe « win-win ») afin de minimiser les résistances (surtout en cas de relocalisations). Ainsi, le discours doit mettre en évidence tous les éléments du risque et pas seulement ses aspects les plus apparents comme les indemnisations. Ces principes sont à la base d'une approche intégrée qui ne se limite pas à des points de vue sectoriels (celui du riverain, de la Ville, du gouvernement).

***En résumé**, une (ou des) stratégie(s) globale(s) combinant les solutions identifiées en pré-faisabilité est recherchée en respectant quelques principes de base comme la rentabilité économique, l'intégration de toutes les composantes du risque et l'équité entre les intervenants. Le critère de réceptivité des solutions par les populations riveraines n'est pas évalué dans le cadre de cette étude.*

3.2.9 Résultats obtenus à Châteauguay

L'application réalisée à Châteauguay a permis de mettre à l'épreuve tous les éléments de la méthodologie proposée. Parmi les innovations, on compte :

1. L'application conjointe du module de glace de HEC-RAS (US Army Corps of Engineers) et de MODELEUR afin de simuler les embâcles historiques et d'analyser l'influence du débit sur le niveau d'eau amont en situation d'embâcle
2. L'analyse de l'impact de l'implantation de structures de rétention de la glace en amont des secteurs vulnérables
3. L'analyse de solutions visant un accroissement de l'hydraulicité du cours d'eau basées sur le rétablissement d'anciens chenaux empiétés par le réseau de la voirie local

4. L'application de méthodes statistiques basées sur les probabilités conjointes afin de déterminer l'effet conjugué de plus d'un facteur de submersion
5. L'inclusion des embâcles dans le calcul de risque
6. La cartographie du risque unitaire d'endommagement (taux d'endommagement) en fonction des divers modes d'implantation résidentielle en zone submersible (ex; hauteur du rez-de-chaussée, présence et état de finition du sous-sol) et des paramètres statistiques et hydrauliques des aléas (crues à l'eau libre et embâcles confondus)
7. L'analyse des opportunités de relocalisations et/ou immunisations
8. L'intégration du schéma de solution proposé dans une vision élargie de la vocation des cours d'eau en milieu urbain (accessibilité au plus grand nombre, par linéaire, etc.)
9. La prise en compte des dommages d'incertitude et leur impact sur la fiscalité municipale, dans une perspective de développement rentable et durable.

Le rapport Leclerc *et al.*, (2006) est en voie d'achèvement et il est appelé à faire école dans le domaine de la prévention des risques d'inondations à cause de la perspective élargie qu'il offre pour le résolution des problématiques les plus lourdes. Le lecteur est renvoyé à ce rapport pour de plus amples détails d'application.

4 Les logiciels

4.1 Modeleur2

Modeleur2 est l'un des logiciels développés lors du projet SEGRI. Il se veut le remplaçant de Modeleur 1.0. Ce logiciel doit prendre la relève en ajoutant toute une gamme de fonctionnalités.

4.1.1 Spécifications fonctionnelles

Afin de produire un logiciel qui réponde aux besoins des utilisateurs, il importe de rencontrer ces utilisateurs et d'aller chercher leurs commentaires et suggestions.

Dans le cas de Modeleur2, cette étape avait été réalisée en 2003 au moyen « d'entrevues » avec les utilisateurs / partenaires. C'est ainsi qu'avaient été rencontrées les personnes suivantes :

- José Bechara (Université Nationale du Nord-Est, Corrientes, Argentine) ;
- Jean Morin, Olivier Champoux, Daniel Rioux (Environnement Canada);
- Paul Boudreau, Michel Leclerc (INRS-ETE);
- Renaud le Boulleur de Courlon (Institut National Polytechnique de Toulouse);
- Jean Gauthier (BPR);
- Richard Frenette (Université d'Ottawa) ;
- Pierre Bélanger (GPR);
- André Robitaille, Michel Carreau (Synexus Global).

Les spécifications fonctionnelles incorporent également toute l'expérience de l'INRS-ETE tant dans l'utilisation de Modeleur que dans la formation des utilisateurs de Modeleur / Hydrosim.

La version finale des spécifications fonctionnelles fut complétée en 2004 (Secretan et al., 2004).

Le logiciel Modeleur2 fonctionne à l'aide d'activités liées typiquement aux tâches de modélisation (voir Figure 2). La plupart des tâches-clés du logiciel peuvent être regroupées sous l'une ou l'autre de ces activités. Bien que chacune de ces activités s'enchaîne aux autres pour proposer un flot de travail à l'usager, très souvent des retours en arrière doivent être effectués pour améliorer ou corriger une donnée. Ces activités sont articulées autour du modèle suivant :

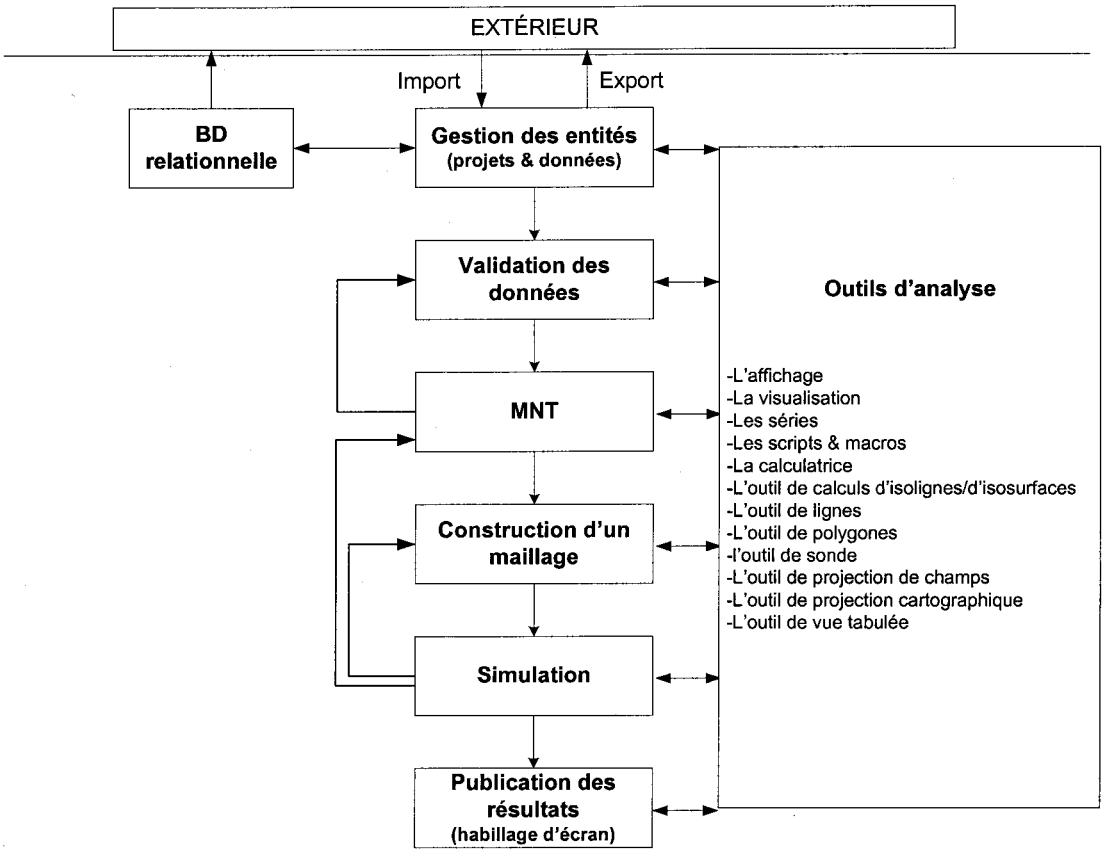


Figure 2 – Flot des activités de Modeleur2

Gravitant autour des activités qui viennent d'être mentionnées, on retrouve une activité d'analyse des résultats qui peut être réalisée avec toute une gamme d'outils d'analyse.

Tout au long des différentes activités, l'usager manipule différents types de données offerts par les « plug-in » (extensions) avec lesquels le logiciel est configuré. La plupart des données de Modeleur2 peuvent être ouvertes, fermées, enregistrées, enregistrées sous et détruites.

L'application Modeleur2 est divisée en trois zones: la zone de contrôle, la zone des fenêtres et la zone d'information (voir Figure 3). Les trois zones peuvent être redimensionnées et réorganisées par l'usager.

La zone de contrôle est une zone à onglets typiquement située à gauche dans la fenêtre principale. Chaque onglet héberge habituellement la zone de contrôle d'un plug-in. Dans la plupart des cas, ceci correspond à un composant de type « arborescence », lequel est utilisé pour explorer et gérer les différentes données offertes par le plug-in.

La zone des fenêtres contient les fenêtres qui ont été créées par l'usager. Il y a également certains plug-in de Modeleur2 qui offrent des fonctionnalités qui mènent à l'ouverture ou à la création de fenêtres.

La zone information se situe au bas de l'écran. Dans cette zone à onglets, les plug-in peuvent ajouter des composants de type informatif. De base Modeleur2 offre un composant de log qui est placé dans cette zone, et également une sonde.

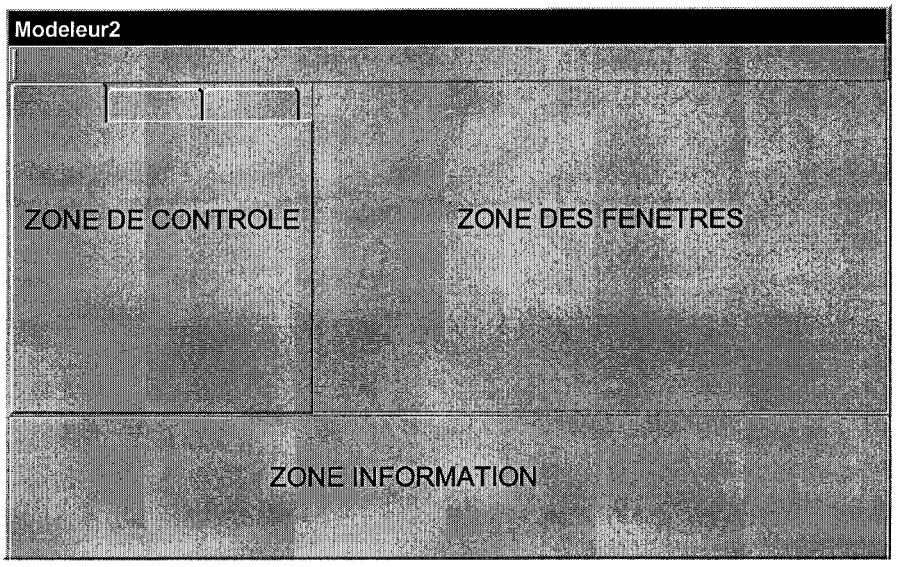


Figure 3 - Interface graphique de Modeleur2

Modeleur2 contient une barre de menus. Les items de menus hébergés dans cette barre peuvent varier en fonction des plug-in présents au démarrage du logiciel. Les plug-in configurent eux-mêmes les entrées de menu requis dans cette barre.

Les prochaines pages proposent un résumé des fonctionnalités proposées par Modeleur2 en reprenant les idées clés de chacune des sections des spécifications fonctionnelles.

4.1.1.1 Activité : projet et gestion des données

Modeleur2 fait appel à la notion de projet pour contenir les données créées ou importées par l'utilisateur. Les données sont toutes emmagasinées dans une base de données dans le « schéma » du projet. De façon simplifiée, un schéma peut être vu comme étant un contenant qui maintient des tables d'informations, chaque donnée étant elle-même modélisée à l'aide d'une ou plusieurs tables.

L'usager peut créer des liens externes vers une ou plusieurs données qui sont hébergées dans un projet autre que celui actuellement ouvert. L'accès aux données par un lien externe est toujours en lecture seule, c'est-à-dire qu'il n'est pas possible de modifier ou de détruire les données d'un projet externe. L'accès aux données peut se faire sur le projet en entier ou seulement sur une partie.

L'usager a accès à une boîte de dialogue qui lui permet d'interagir avec ses données. Une donnée possède de nombreuses métadonnées, dont son nom. L'usager peut créer des liens

externes vers une ou plusieurs données qui sont hébergées dans un projet autre que celui actuellement ouvert.

L'usager peut raffiner la liste des données affichées dans la boîte de dialogue de gestion des données. Ceci lui permet de ne travailler qu'avec les données qui l'intéressent. Il a la possibilité de se définir des critères qui sont en fait des requêtes SQL qui limitent les entrées affichées dans la boîte de dialogue.

Pour l'aider à construire ses critères, l'usager a accès à un outil de construction de requêtes SQL. Cet outil permet à l'usager de construire de telles requêtes (sur les tables de la base de données) sans avoir à connaître en détails la structure de la base de données.

L'usager peut consulter les dépendances des données contenues dans son projet et il peut supprimer une donnée s'il n'y a aucune autre donnée qui dépend d'elle dans le projet.

L'usager peut importer / exporter des données à partir de / vers des fichiers. Pour chaque type de donnée qu'il est possible d'importer / exporter, un format intermédiaire a été défini.

À l'importation, Modeleur2 est en mesure d'amener le contenu de ce fichier au format intermédiaire dans les tables du projet ouvert. L'usager importateur doit au préalable appliquer un prétraitement à ses données pour les traduire dans ce format.

À l'exportation, Modeleur2 est en mesure d'écrire le contenu de ses tables dans un fichier avec le format intermédiaire. L'usager exportateur peut par la suite appliquer un post-traitement pour traduire le contenu de ce fichier dans un autre format dont il est propriétaire. L'usager peut importer / exporter plusieurs fichiers simultanément.

4.1.1.2 Activité : validation des données

Cette activité a pour but de valider les données de terrain qui ont été importées. L'usager interagit avec des points de mesure. Un point possède une coordonnée $\{x,y\}$ et des attributs. Les attributs d'un point sont constitués de variables qui constituent l'information en ce point. Ces variables peuvent être des valeurs numériques, des chaînes de caractères, ou autres.

Quand un point de terrain est importé, on importe du même coup toutes les valeurs de cette donnée $\{x, y\}$, *<attributs>*. L'ensemble des points forme un semi de points de terrain.

Il y a deux types de points : ceux qui proviennent d'une mesure sur le terrain et ceux qui éventuellement ont été ajoutés par l'usager.

La principale tâche que l'usager doit d'abord accomplir est de désactiver certains points aberrants, c'est-à-dire qui ne sont pas cohérents dans son système de validation. Pour ce

faire, il peut faire appel à des filtres pour déterminer les points à désactiver. Les points désactivés ne sont pas effacés (perdus) car l'usager peut en tout temps revenir aux données initiales.

La tâche de filtrage est accomplie en combinant deux types de filtre: les filtres primaires et les filtres secondaires.

Les filtres primaires sont utilisés pour calculer une cote de qualité à chaque point du semi introduit à l'entrée du filtre. Ces filtres peuvent baser leur appréciation sur différents paramètres, et ils attribuent à chaque point du semi une cote entre 0 et 1. La Figure 4 illustre le concept du filtre primaire.

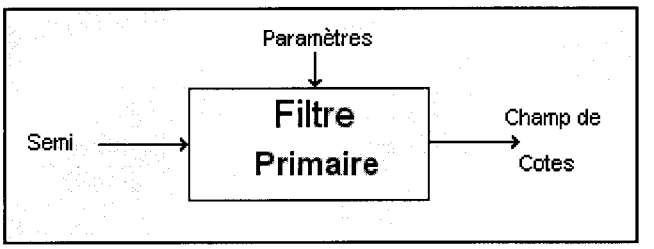


Figure 4 - Filtre primaire

Les filtres secondaires sont les filtres qui vont travailler à partir des cotes produites par les filtres primaires. Ils ont comme rôle de ressortir un champ de booléens déterminé à partir des cotes combinées de filtres primaires et d'une valeur seuil. La Figure 5 illustre le concept du filtre secondaire.

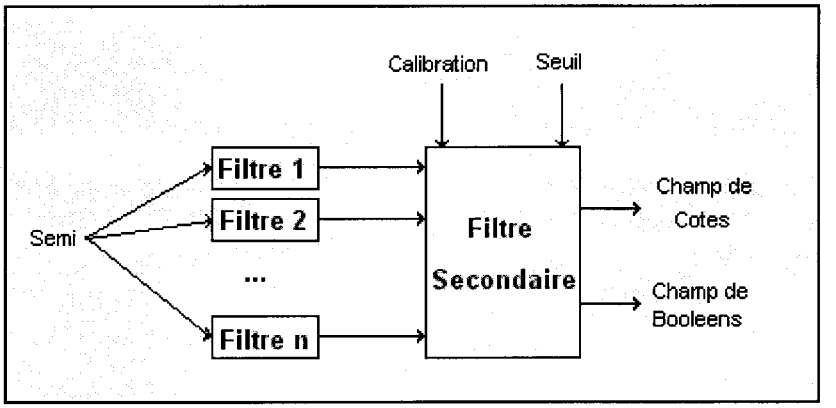


Figure 5 - Filtre secondaire

Les filtres secondaires permettent d'effectuer une pondération pour augmenter / diminuer la valeur relative de l'un des filtres primaires impliqué dans le système. Les combinaisons appliquées peuvent être de simples pondérations ou peuvent être mises en place à partir de techniques plus évoluées comme la logique floue, les réseaux de neurones, les méthodes statistiques, etc.)

Le plug-in de filtrage offert par Modeleur2 permet d'accueillir des filtres conçus par un développeur externe. Ces filtres peuvent être développés en langage C++ ou en Python.

En plus d'utiliser des filtres, l'usager peut également ajouter des points manuellement ou modifier la valeur des attributs de certains points.

4.1.1.3 Activité : publication d'un semi de points

Une fois nettoyé, le semi de points de terrain peut être publié pour devenir un champ qu'on appelle *champ publié*. L'usager pourra utiliser le champ ainsi publié pour monter son MNT (Modèle Numérique de Terrain). Lors de l'opération de publication d'un semi de points, un maillage est créé automatiquement en reliant les points entre eux pour former les éléments du maillage. Le champ publié est évidemment lié à ce maillage.

4.1.1.4 Activité : MNT

Le MNT décrit chacun des aspects du terrain sur lequel l'usager désire baser son étude. Un MNT contient un nombre non déterminé de partitions de données. Une partition est un processus déclaratif permettant d'associer à un domaine ou un ensemble de sous-domaines une valeur ou un jeu de données ou de paramètres quelconque.

Les partitions sont construites en superposant des *couches de données*. Une couche de données est constituée de sous-domaines auxquels est associé un même jeu de données. Un jeu de donnée fait référence à ce que l'on a nommé *champ publié* et correspond en fait à un semi de points qui a été publié à l'activité validation des données.

Les couches de données se voient allouer des ordres de priorité. La couche de données se retrouvant au sommet des priorités (à l'avant-plan) est celle qui sera utilisée lorsque l'information sera transportée (projetée) sur un maillage.

L'usager peut amener une couche en mode d'édition afin d'en modifier le contour. L'usager doit indiquer lorsqu'il désire sortir du mode d'édition. Les modifications faites peuvent alors êtres conservées ou rejetées.

Le ou les sous-domaines d'une couche doivent obligatoirement se trouver dans la limite du jeu de données. La limite du jeu de données correspond au contour du jeu de données, c'est-à-dire les points (x,y) qui délimitent l'enveloppe où le jeu de données peut fournir une valeur par interpolation.

4.1.1.5 Activité : partition de maillage

Cette activité a pour but de construire un maillage d'éléments finis. Pour arriver à compléter cette tâche, l'usager doit se construire une partition de maillage. À l'aide de la partition de maillage, l'usager peut spécifier la géométrie et le degré de raffinement local de son maillage.

La construction des partitions de maillage est basée sur un principe de superposition de couches de maillage en fonction de leur priorité, principe similaire à celui employé pour le MNT.

Une couche de maillage est constituée d'un ou plusieurs sous-domaines qui se voient assignés des paramètres de maillage. Les paramètres de maillage des couches servent à spécifier le degré de raffinement du maillage.

Chaque couche de maillage se voit assigner un ordre de priorité, que l'usager peut changer par la suite. Lorsqu'il y a des zones d'intersection entre deux ou plusieurs couches, les paramètres de la couche avec la plus haute priorité sont employés pour la génération du maillage.

Outre les mailleurs fournis avec Modeleur2, d'autres pourraient être intégrés au logiciel par des développeurs externes. Comme seule contrainte, le mailleur doit être encapsulé dans une DLL (« Dynamic Link Library ») qui réponde à l'interface de mailleur commune à tous les mailleurs. Les mailleurs doivent connaître le domaine à mailler, les contraintes aux frontières et se voir fournir des paramètres de maillage. De base, les paramètres de maillage sont centrés autour du duo de paramètres suivants: un champ d'ellipse erreur et un niveau de tolérance.

Les champs d'erreur peuvent provenir de résultats de simulation (champ d'erreur de niveau d'eau, de débit, etc.), être déterminés à partir de données mesurées (ex : topographie) ou être construits par l'usager (champ analytique).

Avec le niveau de tolérance, le mailleur se charge de mailler la couche en répondant au niveau de tolérance prescrit, c'est-à-dire qu'il ajoute des mailles jusqu'à ce que le niveau d'erreur soit acceptable.

Les données spécifiées lors des différentes activités de construction de partitions (activité : MNT et activité : partition de condition aux limites) peuvent être portées sur un maillage lors de l'activité d'assemblage.

4.1.1.6 Activité : partition de conditions aux limite

L'usager spécifie des conditions aux limites de son domaine. Le modèle de conditions aux limites est spécifié en deux phases. La première phase vise à spécifier la partie « condition » et la seconde vise à spécifier la partie «limites ». Ces deux phases amènent la construction d'une partition de conditions aux limites. L'idée derrière cette séparation est de dissocier le contenu d'une condition de l'arête sur laquelle est utilisée. Lors d'une étude, il arrive fréquemment que la limite physique de la partition change, alors que la valeur de la condition, elle, reste la même.

L'usager peut spécifier la partie condition en ajoutant, enlevant et modifiant des conditions. Ces conditions se voient attribuer un nom, lequel nom est utilisé pour associer

les dites conditions avec les limites de la partition. La Figure 6 fait ressortir la différence entre la phase de définition des conditions et celle de définition des limites.

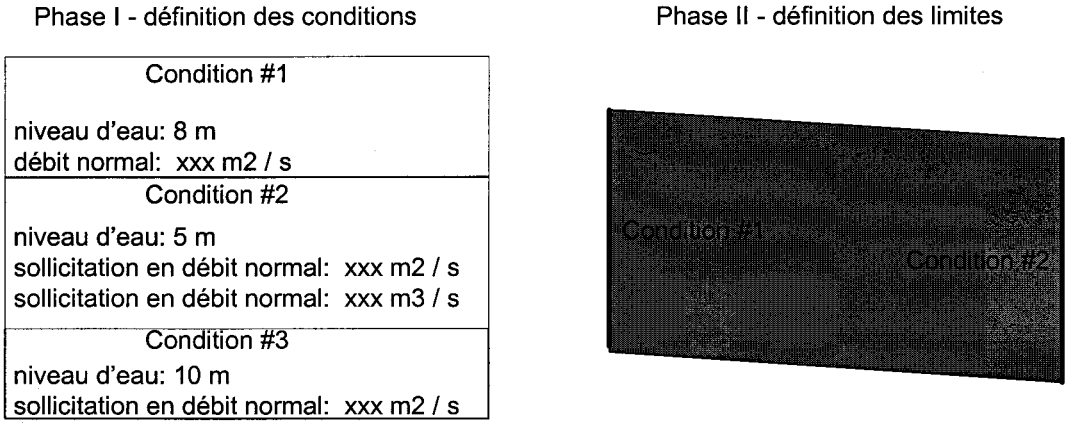


Figure 6 - Partition de conditions aux limites

Les conditions sont bien sûr couplées de très près avec le simulateur hydrodynamique, c'est-à-dire que chaque simulateur (ou plutôt chaque application d'un simulateur) a son jeu de conditions possibles.

Les conditions aux limites du domaine peuvent être spécifiées soit par arête, soit par sommets. Quand les conditions sont spécifiées sur les sommets, il y a interpolation de la condition sur l'arête reliant les sommets. Quand la condition est spécifiée par arête, la valeur est constante partout sur l'arête.

4.1.1.7 Activité : assemblage

L'opération d'assemblage peut se faire sur plusieurs données simultanément. Les partitions d'un MNT peuvent être portées sur un maillage de façon à obtenir des champs pour chacun des aspects du MNT (topographie, substrat, etc.). Pour le MNT, l'opération d'assemblage devrait pouvoir s'exécuter avec succès si le maillage a été construit en respectant les limites du squelette du MNT (plus petite zone commune à toutes les partitions).

Les partitions de conditions aux limites, elles, peuvent être portées sur un maillage pour produire des conditions aux limites.

L'usager choisit d'abord son maillage. Il choisit ensuite les partitions qu'il désire assembler. Finalement, il attribue un nom à chacune des données qui résultera de l'opération d'assemblage. L'usager peut spécifier ces informations pour tous les champs qu'il désire assembler et lancer l'opération d'assemblage en une seule étape.

Si, pour une raison ou une autre, il y a une erreur lors de l'assemblage d'une ou plusieurs partition, l'usager peut aller corriger sa partition ou son maillage et répéter l'opération sans que la boîte de dialogue n'ait été fermée. L'usager se voit fournir de l'information

pouvant l'aider à identifier le problème rencontré. Les données qui ont été assemblées correctement sont enregistrées dans la base de données et peuvent être consultées. Les données qui ont été assemblées avec succès sont marquées d'une façon différente de celles pour lesquelles il y a eu une erreur lors de l'assemblage.

4.1.1.8 Activité : simulation

Une simulation est une suite d'étapes de calcul (ex : hydrodynamique) qui a pour but d'obtenir une solution finale satisfaisante. Chaque calcul est réalisé à l'aide de paramètres d'entrée et produit un ensemble de résultats (ex : champ de niveaux d'eau et champ de vitesses). Les résultats d'une étape de calcul peuvent à leur tour servir de paramètres d'entrée (i.e. solution initiale) à une autre étape de calcul.

Une simulation n'est pas nécessairement continue, c'est-à-dire que le chemin à suivre pour arriver à la solution finale n'est pas nécessairement exempt de retours en arrière. Pour arriver à gérer ceux-ci, l'usager a accès à un historique sous forme d'un graphe présentant toutes les étapes de calcul qui furent suivies.

Si, pour une raison quelconque, un chemin de simulation ne mène nul part, il est possible pour l'usager de repartir à une étape de calcul précédente. De là, il est possible de revisiter les paramètres d'entrée de ce calcul et de le reprendre. Il est également possible de reprendre le calcul à partir de toutes nouvelles conditions initiales.

Un des objectifs poursuivis est de pouvoir analyser une simulation et ses étapes de calculs clés afin de déceler les raisons d'une divergence possible. Un autre objectif est de pouvoir aider les usagers externes à distance en analysant leur simulation.

Un calcul de simulation est réalisé à l'aide d'un simulateur. Un simulateur peut être vu comme une boîte noire qui reçoit des paramètres en entrée et qui en ressort des résultats. Les simulateurs doivent fournir à Modeleur2 un moyen d'exporter les données d'entrée au simulateur et d'importer les résultats vers Modeleur2.

Le même simulateur peut être utilisé de plusieurs manières différentes, c'est-à-dire dans plusieurs types d'application. Par exemple, H2D2, le simulateur de base fourni avec Modeleur2, peut être utilisé pour faire des calculs hydrodynamiques, des calculs de température, des calculs de transport-diffusion, etc. Chaque utilisation d'un simulateur peut demander des paramètres d'entrée et produire des résultats différents de l'un à l'autre.

Les simulateurs sont des plug-in que l'on peut rattacher à Modeleur2. Le logiciel offre tous les mécanismes pour pouvoir intégrer plusieurs simulateurs, et n'est donc pas spécifiquement dédié à un simulateur particulier.

Les simulateurs peuvent être contrôlés pour effectuer des calculs à distance. Pour cela, une configuration doit être faite pour faire connaître les machines distantes à utiliser et leurs paramètres.

Des mécanismes sont également mis en place pour que chaque simulateur puisse faire retourner des informations sur l'avancement de son calcul. L'usager a donc accès à cette rétroaction (*feedback*) en temps réel lors de l'exécution d'un calcul.

4.1.1.9 Activité : publication des résultats

L'usager a accès à des fonctionnalités pour ajouter des composants visuels à son affichage pour fin de présentation (publication) par exemple.

4.1.1.10 Outil : affichage

Un *espace de représentation* correspond au contenu d'une fenêtre graphique. Un espace de représentation est composé de trois plans : plan arrière, plan médian et plan interactif. Un plan peut héberger un nombre indéterminé d'objets graphiques.

Un objet graphique correspond à la représentation visuelle d'une donnée. Il possède les moyens de convertir en primitives graphiques la donnée à laquelle il est lié. Ces primitives graphiques peuvent ensuite être envoyées à l'affichage. Un objet graphique possède les attributs suivants: *affichable*, *imprimable* et *une échelle de couleurs*. Ces attributs peuvent être modifiés par l'usager. L'usager interagit avec une liste d'affichage qui contient tous les espaces de représentation qui ont été créés (i.e. toutes les fenêtres). L'usager peut y voir les objets graphiques contenus dans chaque plan et interagir avec eux.

L'usager a accès à des fonctionnalités pour gérer la caméra. Il peut ainsi effectuer des opérations de zoom-in/zoom-out manuelles ou activer un mode interactif qui donne l'équivalent du mode « loupe » dans certains logiciels. Il peut se déplacer dans sa fenêtre graphique en effectuant des translations ou de façon interactive (panoramique) à l'aide de l'équivalent de « la petite main », façon de faire très utilisée dans les logiciels de dessin. Enfin, l'usager peut demander à ce que la zone d'affichage soit remise à la valeur par défaut qui est spécifiée dans les paramètres du projet.

4.1.1.11 Outil : visualisation

L'usager peut visualiser un nouvel objet graphique de type maillage en sélectionnant un maillage parmi la liste des maillages connus du projet. L'objet graphique est construit à partir de paramètres fournis par l'usager et peut ensuite être affiché.

L'usager peut sélectionner une couleur constante pour son maillage. Il peut également faire porter de l'information par la couleur. Pour cela, l'usager doit spécifier un champ scalaire d'où est extraite l'information sur la couleur. L'usager peut aussi faire porter un champ sur la coordonnée « z » (ex : champ de niveau d'eau) ce qui permet d'afficher le maillage en trois dimensions.

En spécifiant un champ scalaire, il est également possible de demander l'affichage des isolignes et d'isosurfaces à l'engin graphique derrière Modeleur2.

L'usager peut visualiser des points qui sont spécifiés par une coordonnée x et une coordonnée y. Il a plusieurs façons de faire pour générer les points à tracer, et pour chacune d'elles, il y a différentes façons de faire porter de l'information sur les points.

L'usager peut générer les points à tracer à partir des nœuds d'un maillage, en itérant sur une région ou à partir d'un semi de points.

4.1.1.12 Outil : construction de série

Une série est un ensemble de coordonnées/valeurs que l'usager utilise pour obtenir une valeur interpolée pour n'importe quelle coordonnée dans la région couverte par la série. Sous Modeleur2 les valeurs peuvent être des réels ou des champs. Le concept de série de même que la façon dont il est implanté dans Modeleur2 sont décrits en détails dans la section 4.1.2 qui traite de l'architecture du logiciel (voir la section Les séries).

Pour construire sa série, l'usager doit spécifier les couples coordonnées/valeurs qui la caractérise. La méthode employée pour la construction des séries est basée sur une grille régulière.

Pour définir l'espace de solution, l'usager a la possibilité de construire une *grille régulière* simple en fournissant une coordonnée d'origine et un pas de coordonnée constant entre les nœuds de chacun des axes. L'usager a également la possibilité de construire une grille régulière plus complexe en spécifiant un pas de coordonnée variable pour chaque axe. La spécification de la coordonnée sur chaque axe se fait à l'aide d'un tableau qui indique la coordonnée sur l'axe en fonction de l'indice. La Figure 7 propose un exemple de grille régulière typique.

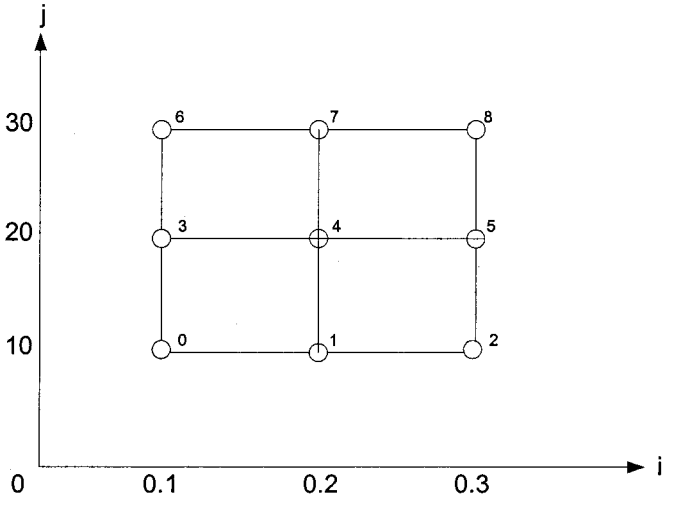


Figure 7 - Grille régulière d'une série

Une fois la grille régulière spécifiée, l'usager peut procéder à l'assignation des valeurs. Les valeurs de l'espace de solution doivent toutes être du même type (condition d'homogénéité), par exemple des champs de niveaux d'eau issus de simulations sur le même maillage éléments finis. La Figure 8 illustre l'étape d'assignation des valeurs de la série.

Coordonnées	Indice nœud	Valeur
i = 0.1, j = 10	0	Champ niveau d'eau #1
i = 0.2, j = 10	1	Champ niveau d'eau #2
i = 0.3, j = 10	2	Champ niveau d'eau #3
i = 0.1, j = 20	3	Champ niveau d'eau #4
i = 0.2, j = 20	4	Champ niveau d'eau #5
i = 0.3, j = 20	5	Champ niveau d'eau #6
i = 0.1, j = 30	6	Champ niveau d'eau #7
i = 0.2, j = 30	7	Champ niveau d'eau #8
i = 0.3, j = 30	8	Champ niveau d'eau #9

Figure 8 - Spécification des valeurs d'une série

4.1.1.13 Outil : scripts et macros

L'outil de script a pour but de permettre à l'usager d'automatiser le fonctionnement de Modeleur2 et de fournir l'accès à la plupart de ses fonctionnalités « cachées ». L'usager peut ainsi créer des scripts en Python qu'il sera ensuite capable d'exécuter.

Pour écrire un script, l'usager a deux choix : soit il édite un *script* en spécifiant (rédigeant) lui-même le contenu, soit il crée une *macro*. Une *macro* est un script qui enregistre automatiquement toutes les actions produites dans Modeleur2 pendant une période de temps définie par l'usager. Pour que l'usager puisse éditer un script, Modeleur2 intègre un IDE (Integrated Development Environment) externe contenant un éditeur.

4.1.1.14 Outil : calculatrice

Modeleur2 offre aux usagers une *calculatrice* qui permet de manipuler des types de données simples (ex : entier ou double précision), mais également de manipuler certaines structures de données de Modeleur2 (ex : les champs et les séries). La calculatrice sous Modeleur2 utilise l'interpréteur Python comme plaque tournante.

Le calculatrice inclut un clavier ayant une apparence similaire à celui de la calculatrice sous Windows. Le clavier de la calculatrice est là pour éviter d'obliger l'usager à rédiger du code (Python) manuellement. On veut éviter le « syndrome de la page blanche » car

beaucoup d'usagers ne sont pas à l'aise avec l'idée de devoir programmer. Il faut donc les placer dans un contexte de fonctionnement convivial auquel ils sont plus habitués et dans lequel il pourront effectuer un travail plus efficace.

4.1.1.15 Outil : sonde

Cet outil permet d'obtenir la position géographique d'un point et d'interpoler les valeurs d'un ou plusieurs champs ou série à cette position géographique.

La sonde peut être utilisée dans n'importe laquelle des fenêtres graphiques de Modeleur2. La sonde permet de voir les coordonnées en fonction de la position de la souris dans la fenêtre graphique.

4.1.2 Architecture

Dans le processus de ré-ingénierie de Modeleur2, d'importantes décisions ont dû être prises sur le plan technique, souvent des décisions qui allaient dicter la suite du développement.

Parmi ces décisions, on peut en identifier certaines qui ont eu un impact majeur sur le logiciel en entier, notamment :

- L'utilisation d'un bus d'événements (communication entre modules);
- L'utilisation d'une base de données relationnelle (avec support spatial);
- L'utilisation de VTK et MayaVi pour les représentations graphiques;
- L'utilisation de Python comme langage de script;
- L'architecture avec plug-in;
- La mise en place du concept de séries.

4.1.2.1 L'utilisation d'un bus d'événements

L'architecture de Modeleur2 est centrée autour d'un bus d'événements qui a comme fonction de transmettre des événements entre les modules du système.

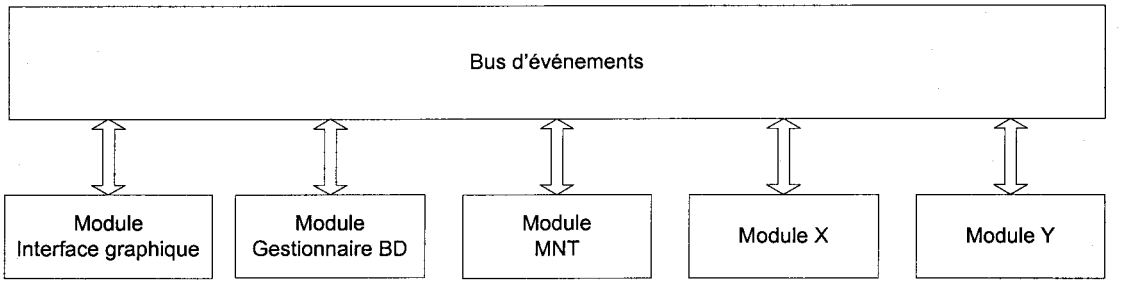


Figure 9 - Bus d'événements

Les raisons suivantes justifient le choix du bus logiciel comme plaque tournante du logiciel:

- Le bus d'événements permet de minimiser le couplage (interaction directe) entre les différents modules. Il est donc possible d'en enlever et d'en ajouter sans répercussions indésirables sur le système en entier.

- Il est possible d'interchanger un module par un autre à condition qu'ils répondent tout deux aux mêmes événements.

- Il est possible d'enregistrer et d'exécuter des macros en étudiant le flot d'événements et en le reproduisant lors d'une nouvelle exécution.

- Les différents types de liens permettent une extensibilité multi-tâches ou multiprocesseurs.

- La possibilité de construire toute une gamme de logiciels « différents » en déterminant les modules à ajouter.

Voici une courte définition des termes qui sont reliés au bus logiciel :

***Bus d'événements**:* mécanisme ayant pour but d'effectuer la distribution d'événements aux différents modules qui y sont reliés. Le bus d'événements contient une liste des modules et distribue les événements à chaque module dans cette liste.

Module: ensemble de fonctionnalités logicielles, connectées au bus d'événements afin de pouvoir y envoyer des événements et en recevoir. Cet ensemble est représenté par un objet qui possède une méthode spécialisée dans la réception d'événements et a accès à une méthode d'envoi d'événement.

Lien : intermédiaire liant le bus et un module. Le lien peut être de différents types : direct, par tâches, distant, etc. La notion de lien permet de garder un faible couplage entre le bus et le module. Ceci permet d'ajouter un type de lien sans avoir à modifier le bus ou les modules.

***Événement*:** unité d'échange sur le bus d'événements. Un événement peut contenir à la fois de l'information, une commande ou encore rien.

L'exécution d'une application utilisant le bus d'événements peut se diviser en différentes étapes. Tout d'abord, l'objet « bus d'événements » et tous les modules sont créés. Ensuite, un lien est construit pour chaque module, lien qui est constitué de quatre connexions (voir Figure 10).

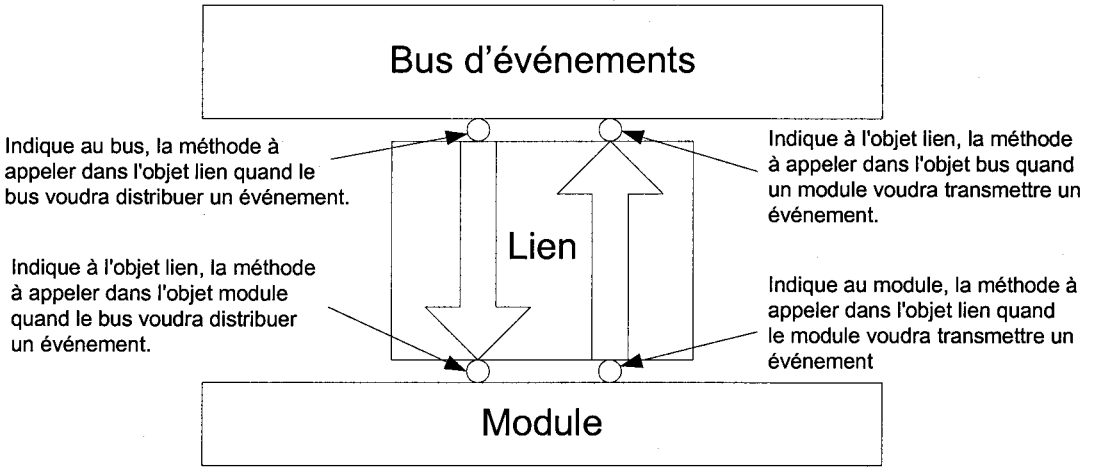


Figure 10 - Fonctionnement du bus d'événements

A la fin de cette phase, chaque module créé a le pouvoir d'envoyer et de recevoir des événements par l'intermédiaire du bus.

Une fois le bus initialisé, tout objet ayant accès au bus peut lancer des événements. Ces événements sont reçus par le bus. Le bus distribue les événements à chaque module de sa liste de modules connectés. La distribution s'arrête lorsque la fin de la liste est rencontrée ou lorsque que l'un des modules retourne un message d'erreur.

À la fermeture de l'application, il est nécessaire d'avertir les différents modules ouverts *via* un événement de fermeture. Ceci doit être fait afin de s'assurer qu'ils sont prêts à être fermés. Suite à l'événement de fermeture, certains modules peuvent à leur tour envoyer un événement sur le bus. Par exemple, un module pourrait envoyer un événement visant à demander à l'usager s'il désire enregistrer son travail avant de quitter.

Les événements sont à la base de la communication dans le modèle "Module-Événements"; il s'agit en fait de petites classes C++ dérivant d'une classe de base ("l'événement de base").

Les événements sont divisés selon une hiérarchie de classes parents abstraites héritant toutes de l'événement de base. Un événement peut contenir tout type de données et sera transmis tel quel sur le bus. En fait, le bus donne un pointeur vers l'événement au module récepteur; c'est le même événement qui est donné successivement a chaque récepteur. Tout module récepteur sera à même d'accéder à l'événement et d'y appeler les méthodes de requête et d'assignation voulues.

Les échanges s'effectuent via un mécanisme de callback (mécanisme de rappel). Lors de la création d'un module, celui-ci se voit assigné un callback lui permettant d'envoyer de nouveaux événements au bus. Il y a également création d'un callback vers la méthode de réception d'événement du module (*recoisEvenement()*). Ce callback est alors assigné au bus.

À l'intérieur d'un module, il sera donc possible d'utiliser en tout temps le "callback" vers le bus d'événements pour envoyer des événements. L'appel de cette méthode sera fait avec un événement en paramètre.

L'approche « par module » permet de circonscrire clairement le rôle à jouer par chacun des modules, tout en gardant le minimum de dépendances avec les autres modules. Elle permet le développement de plusieurs modules de manière concurrente une fois l'interface « événementielle » déterminée. Ceci est très approprié à la définition des mandats de stage, car les liens de dépendance entre les tâches sont minimes. Il est également plus aisé de corriger les bogues trouvés dans un module sans avoir à toucher au reste du logiciel.

4.1.2.2 L'utilisation d'une base de données relationnelle

Modeleur2 est basé sur un SGBD (Système de Gestion de Base de Données) relationnel. Le fait de faire appel à une base de données relationnelle est un changement majeur qui affecte le logiciel en entier.

Implanter le modèle de données orienté objet de Modeleur2 dans une base de données relationnelle ne fut pas une tâche triviale. Le fait de mettre en place une architecture où le traitement BD est réparti entre les plug-in a ajouté un niveau de complexité supplémentaire à la tâche.

Beaucoup de temps a donc été investi pour arriver à construire un modèle robuste, complet et flexible qui représente bien les données de Modeleur2. Les avantages et les nouvelles fonctionnalités disponibles dans le logiciel justifient tous les efforts qui ont été mis pour construire ce système.

Voici une liste des fonctionnalités et des gains réalisés en comparaison avec la base de données hybride/relationnelle au cœur de Modeleur 1.0.

- L'accès aux données peut se faire directement à la base de données (via le langage SQL) plutôt qu'avec des filtres, seul moyen d'accéder aux données dans Modeleur 1.0. Sous Modeleur2, les usagers ont accès aux tables et à la structure de la base de données. Ils peuvent ainsi accéder aux données d'une façon tout à fait standardisée (connexion ODBC et langage SQL).

- La cohérence et l'intégrité des liens entre les données Modeleur2 est automatiquement vérifiée. La base de données ne permet pas de créer des incohérences puisqu'elle vérifie les contraintes entre les tables d'information à chaque tentative d'insertion, de suppression ou de mise-à-jour.

- La présence de mécanismes de transaction et de retours en arrière (« rollback ») permet de bien gérer l'unicité d'une action qui s'effectue sur plusieurs tables en laissant la BD dans son état initial si une erreur survient.

- Plusieurs utilisateurs peuvent accéder à la même base de données simultanément. La base de données gère l'unicité des transactions de façon à ce que les utilisateurs aient une image du contenu de la BD cohérente et juste en tout temps.

- Les utilisateurs peuvent se voir octroyer des droits qui peuvent limiter les accès en écriture, lecture et modification sur certaines tables ou schémas. Il est ainsi possible de gérer à partir d'une BD centralisée du contenu à partager (ex : projets modeleur2) mais où seuls les administrateurs peuvent ajouter du contenu.

- Il est possible de faire des requêtes sur n'importe quel attribut de n'importe quelle table de la base de données. Il est ainsi possible d'extraire de l'information de la BD avec tout type de critères, plutôt qu'un jeu de critère limité comme dans Modeleur 1.0 (date, heure, type, nom).

- La plupart des SGBD offrent des moyens d'effectuer des opérations de sauvegarde / récupération (back up/restore) sur une partie ou la totalité d'une base de données.

- Le support SIG (support standard pour les données de type géographique) ajoute la possibilité de faire des requêtes spatiales. C'est un aspect clé de Modeleur2 car cela permet de délimiter l'espace de travail quand on a affaire à des jeux de données de grande taille.

Modeleur2 peut fonctionner localement si un SGBD est installé sur le poste de l'utilisateur. Le logiciel peut aussi fonctionner en se connectant à une base de données distante via un lien ODBC.

Le SGBD utilisé pour le développement de Modeleur2 est PostgreSQL, et son extension PostGIS qui permet de manipuler les objets SIG.

Bien qu'aucun test ne fut réalisé, d'autres SGBD devraient être supportés avec peu ou pas de modifications, car le développement s'est fait en faisant appel à des éléments standards à tous les SGDB. En théorie, les usagers pourront donc opter pour la base de données qui leur convient.

La base de données a été développée à l'aide du logiciel Case Studio 2 qui permet de générer des scripts SQL pour construire la base de données. Ces scripts peuvent être générés et optimisés pour chaque SGDB.

Pour interagir avec la base de données de manière transparente, un lien ODBC est utilisé. Le développement du code interagissant avec la base de données s'est fait avec la librairie OTL qui permet d'effectuer la communication *via* des « streams » (flots) en encapsulant le lien ODBC.

4.1.2.3 L'utilisation de VTK pour les représentations graphiques

Afin de permettre une expansion plus aisée des capacités de représentation graphique, le moteur graphique de Modeleur a été changé. Le moteur graphique de Modeleur2 est basé sur VTK (Visual ToolKit). Cette librairie graphique est relativement de haut niveau et, lorsque disponible, s'appuie sur des composantes OpenGL.

Cette librairie réponds aux critères suivants qui ont justifié son choix: gratuité, indépendance de plateforme et du système de fenêtrage, fonctionnalités avancées, disponibilité sous Windows et support du 2D/3D.

VTK est basé sur le paradigme par flux de données. Dans ce paradigme, les données circulent dans un réseau de modules. Ces modules sont en fait des algorithmes qui modifient les données qui entrent dans ceux-ci. La Figure 11 illustre la structure typique d'un programme utilisant VTK (les éléments en pointillé dans le schéma sont facultatifs) :

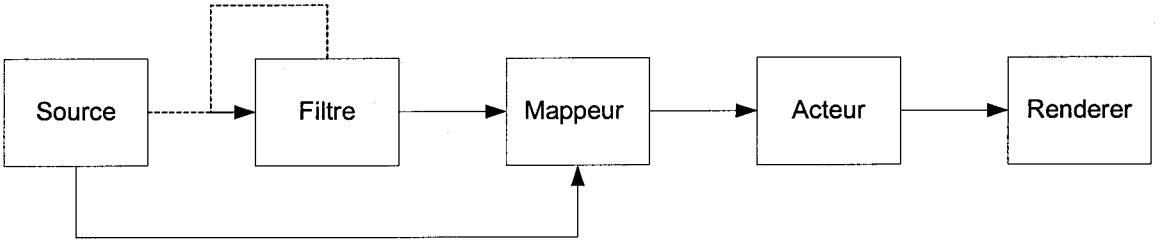


Figure 11 - Pipeline VTK

Voici la signification de chacune des composantes du « pipeline » VTK :

Source :	correspond à la source des données caractérisant l'objet à afficher (ex : un nuage de points)
Filtre	correspond à un filtre qui traite les données qui y entrent pour obtenir des données modifiées en sortie (ex : triangulation sur un nuage de points)
Mappeur	sert à interpréter les données qui entrent en primitives graphiques comme des points, des lignes et des polygones (ex : tracé d'isosurfaces)
Acteur	correspond à l'élément de la scène. C'est avec ce dernier qu'il est possible d'interagir pour positionner l'objet à afficher et en modifier la couleur ou l'opacité.
« Renderer »	est la scène contenant tous les acteurs. Il s'agit ici d'une vue (« viewport ») qui sera contenue dans une fenêtre. Une fenêtre peut ainsi contenir plusieurs « renderers ».

Lors des deux premières années du projet, un modèle basé sur la notion d'objets graphiques a été mis en place. Un objet graphique contient ou un plusieurs acteurs VTK de même que les attributs qualifiant cet objet (ex : échelle de couleur, affichable ou non, etc.). L'objet graphique est envoyé au module d'affichage qui se charge d'ajouter les acteurs VTK et d'afficher la représentation graphique dans la fenêtre appropriée.

Cette approche permet d'avoir le contrôle complet sur qu'on affiche, mais en contrepartie, cela exige de développer des algorithmes de tracé pour remplir les acteurs VTK. Entre autres, les représentations graphiques des objets géométriques sont réalisées avec cette architecture. Initialement, les opérations de tracés étaient également implantées de cette manière.

Toutefois, dans la dernière année du projet, MayaVi, un outil externe qui permet d'effectuer une partie du travail VTK a été intégré à Modeleur2, venant du même coup élargir de façon imposante les capacités offertes par le logiciel. Les opérations de tracés (ex : maillages, iso lignes, isosurfaces, etc.) allaient désormais être confiées à ce composant externe.

Le prix à payer pour bénéficier de cette innovation est d'inverser la façon de faire des opérations de tracés comparativement à Modeleur 1.0. Par exemple, dans Modeleur 1.0, l'usager pouvait demander un tracé d'isosurfaces en fournissant un champ, alors qu'avec

MayaVi, il doit charger d'abord les données (ex : champ) et ensuite il peut demander une tracé d'isosurfaces.

MayaVi se place au dessus de VTK et permet de gérer une partie des étapes mentionnées dans le diagramme (il gère les étapes: Filtre, Mappeur et Acteur). Il s'intègre à Modeleur2 dans la zone de contrôle au même titre qu'un autre plug-in et rend disponible une gamme étendue de fonctionnalités permettant d'effectuer des représentations graphiques variées.

Via une boîte de dialogue qui a été développée à cet effet, MayaVi permet de charger des données provenant de Modeleur2, notamment les maillages et les champs et d'y appliquer toute une gamme de traitements.

Par exemple, il est possible d'appliquer des filtres sur les données fournies en entrée (ex : effectuer une triangulation sur un nuage de points), puis de demander de tracer des isosurfaces ou des iso lignes à partir des données filtrées. Enfin, le résultat peut être affiché avec les couleurs choisies toujours via MayaVi.

MayaVi donne accès à tous les paramètres qui sont disponibles pour configurer chacune des étapes du pipeline et chacun des objets VTK.

Dans un premier temps, un guide d'utilisation des fonctionnalités à utiliser pour effectuer des affichages de base sera disponible aux usagers. L'utilisation des autres fonctionnalités sera laissée à la discrétion des utilisateurs experts qui veulent tirer profit de la présence de VTK. L'important est de prendre conscience de toute la puissance qu'on offre aux utilisateurs avec la combinaison VTK/MayaVi.

4.1.2.4 L'utilisation de Python comme langage de script

L'utilisation de Python dans Modeleur2 comble cinq besoins clés :

- servir à construire l'interface graphique de Modeleur2;
- devenir le remplaçant du langage de la calculatrice de Modeleur 1.0;
- agir comme pièce centrale du système de plug-in (ouverture vers l'extérieur);
- servir de moyen pour tester le logiciel par envoi d'événements;
- permettre d'enregistrer et de rejouer les événements sur le bus (macros).

Servir à construire l'interface graphique de Modeleur2

L'interface graphique (GUI) est entièrement pilotée à partir de Python/wxPython. Elle est interprétée à l'exécution et peut donc être modifiée simplement en modifiant un des scripts qui la compose sans qu'on ait à recompiler quoi que ce soit.

L'interface graphique de Modeleur2 n'est pas centralisée car tous les plug-in peuvent contenir une partie GUI codée en Python et venir la greffer à l'interface graphique principale de Modeleur2.

Il est pertinent de mentionner que le langage wxPython offre toutes les fonctionnalités pour construire une application professionnelle et standardisée. Enfin, il est bon de souligner que des outils sont disponibles pour construire cette interface graphique (ex : Boa Constructor).

Devenir le remplaçant du langage de la calculatrice de Modeleur 1.0

Le passage d'un langage de programmation à l'autre n'est jamais facile. Profitant du fait que Python est lui-même écrit en C, des outils comme SWIG et Boost-Python ont été développés. Ils permettent de « traduire » du code C++ et de le rendre disponible en Python.

La publication des structures de données de Modeleur2 en Python permet une évolution majeure par rapport à Modeleur 1.0. En effet, sous Modeleur 1.0, seuls les champs éléments finis étaient disponibles dans la calculatrice. Sous Modeleur2, c'est toutes les structures de données que l'on rend disponible en Python (événements, champs, maillage, séries, partitions, etc.).

Évidemment, toute cette souplesse a un prix. En effet, il est important de mentionner que la traduction de structures C++ en Python de la complexité de celles contenues dans Modeleur2 n'est pas une tâche triviale, loin de là. Ceci a tout de même pu être réalisé avec succès, ce qui permet de libérer une puissance inégalée en ce qui a trait à la flexibilité et aux usages qui peuvent être faits du logiciel.

Agir comme pièce centrale du système de plug-in

L'utilisation du langage Python est la plaque tournante de l'architecture avec plug-in (décrite dans la section suivante). Sa grande flexibilité a permis de mettre en place un système qui permet d'ajouter des composantes via un script de configuration. La plupart des modules offerts dans Modeleur2 sont d'ailleurs des plug-in qui peuvent au besoin être activés ou désactivés (via le script de configuration).

Le choix d'utiliser Python a également permis d'intégrer au logiciel des outils externes, notamment : MayaVi, Boa Constructor, Winpdb. Le fait que Python soit un langage interprété ouvre la porte à toute la flexibilité qui était souhaitée.

Tester le logiciel par envoi d'événements

Le banc de test automatisé est construit en envoyant sur le bus des événements pour « imiter » les opérations faites par l'interface graphique. Ceci est réalisable car tous les événements ont été publiés en Python et sont donc disponibles autant pour la partie GUI des modules que pour l'écriture de scénarios de test. Grâce à la publication des événements en Python, il fut possible de tester de façon automatique toutes les fonctionnalités du logiciel.

Enregistrer et rejouer les événements sur le bus (macros)

En plus de toutes ces fonctionnalités, le langage de script est utilisé pour enregistrer dans un fichier (i.e. un script Python) les événements qui circulent sur le bus. Il est alors possible de rejouer la suite d'événements. Ceci permet d'implanter une fonctionnalité de macros avec laquelle les usagers peuvent répéter une séquence d'actions automatiquement.

4.1.2.5 L'architecture avec plug-in

Un *plug-in* est une extension d'un logiciel permettant d'ajouter des fonctionnalités à celles qui sont offertes dans sa version de base. Modeleur2 pousse plus loin cette notion en considérant comme « plug-in » tout ce qui n'est pas essentiel à son fonctionnement « minimal ». Le logiciel peut être vu comme une charpente à laquelle on peut ajouter des fonctionnalités.

Dans Modeleur2, un plug-in est composé d'une ou plusieurs DLL, d'un ou plusieurs modules Python, de l'aide et de fichiers de langues. Certains modules ont besoin d'interagir (interfacer) avec l'interface graphique (GUI ou Graphical User Interface). Ceux-ci ont la possibilité d'ajouter des contrôles dans la zone de contrôle, d'ajouter des menus dans la zone de menus, d'ajouter une barre d'outils dans la barre d'outils principale du logiciel, et de s'enregistrer pour l'écoute d'événements provenant du C++ recevoir des événements du C++.

L'usager se voit offrir le moyen de configurer les plug-in à connecter au bus au moyen d'un script de démarrage. Pour ce faire, l'usager doit encapsuler son module dans une DLL. Un module en C++ doit dériver du module de base (`TEModuleBase`). Il doit également spécialiser la fonction virtuelle d'envoi et de réception d'événements. Une fois ces étapes réalisées, le module se voit octroyer les capacités pour transmettre des événements sur le bus et en recevoir. Le module de l'usager peut donc interagir avec tout le reste du système *via* l'échange d'événements.

L'usager peut ajouter ses événements au système en respectant l'interface C++ prescrite. L'usager doit également publier ceux-ci dans Python. Les événements doivent être publiés dans Python car le module de macro doit être en mesure de capter et reproduire tous les événements qui circulent sur le bus.

Certains plug-in offrent la possibilité de créer, charger, enregistrer et supprimer des données dans la base de données. Pour cela, ils doivent aller inscrire des algorithmes BD (GDAlgoIo) pour effectuer ces opérations. Les algorithmes incluent entre autres des requêtes SQL pour effectuer la lecture, l'écriture ou la suppression dans une base de données. Tout plug-in qui désire utiliser les fonctionnalités offertes par ces algorithmes y aura accès car elles sont disponibles à tous.

Les plug-in qui le désirent peuvent venir ajouter des menus à la barre de menus de Modeleur2. Ils peuvent ainsi rendre disponibles leurs fonctionnalités au monde extérieur.

Tout plug-in intéressé à ajouter des menus doit indiquer au gestionnaire de menus quels sont les menus / sous-menus qu'il désire ajouter. Le plug-in doit également spécifier à quel endroit (chemin) dans l'arborescence de menus ceux-ci doivent être ajoutés. Enfin, le plug-in doit associer un callback (fonction à appeler dans le script Python) à chaque entrée de menu qu'il ajoute. Quand l'usager clique sur ce menu, le callback est appelé et le travail s'effectue.

Le plug-in doit également être en mesure de répondre à un événement qui est régulièrement envoyé par l'interface graphique afin de savoir quelles commandes doivent être actives. Le plug-in doit alors indiquer toutes les commandes qu'il désire rendre disponibles à l'usager (entrée de menu non grisée dans la barre de menu).

L'interface graphique (GUI) de Modeleur2 est constituée de trois zones. Deux de ces zones sont des zones de contrôle, c'est-à-dire que des contrôles (ex : arborescence) peuvent y être ajoutés. Chaque module peut inscrire les contrôles qu'il désire dans le gestionnaire de contrôles. Quand un plug-in a besoin de ce contrôle, le gestionnaire le construit et l'ajoute dans la zone de contrôle.

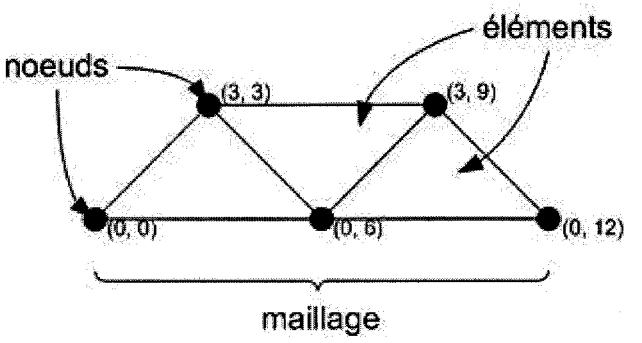
Le GUI de Modeleur2 est constitué d'une zone de barres d'outils. Dans cette zone, les plug-in peuvent insérer une barre d'outils. L'état des boutons de la barre d'outils doit être géré par les plug-in.

La partie GUI de certains plug-in doit parfois être en mesure de recevoir des événements du code C++. Ces plug-in doivent donc s'inscrire comme « écouteur » pour un type d'événement particulier. Quand le gestionnaire d'événements recevra ce type d'événement, il appellera tous les callbacks qui auront été enregistrés par les plug-in.

4.1.2.6 Les séries

La *série* est une structure de données avec laquelle les utilisateurs de Modeleur2 vont interagir. Dans le présent contexte, cette série est essentiellement basée sur les structures de maillage et de champ éléments finis, deux structures de données fondamentales dans Modeleur2 qui permettent une représentation numérique des données de terrain et de simulation. Pour cette raison, il apparaît pertinent d'effectuer le rappel suivant au sujet des éléments finis.

La *méthode des éléments finis* est une technique d'interpolation qui consiste à discrétiser le domaine de calcul en un ensemble de sous domaines de géométrie simple, les éléments. Sur chaque élément, on définit une fonction d'interpolation qui prend en compte la disposition des nœuds de l'élément (géométrie) et les valeurs qu'ils portent.



L'ensemble des nœuds et des éléments forme un maillage. Le champ éléments finis est la structure de données qui rassemble les valeurs nodales sur un maillage éléments finis. Une valeur en tout point de la région spatiale du champ est obtenue par interpolation au sein des éléments.

La différence entre un champ éléments finis conventionnel et la série telle que définie ici se situe au niveau du domaine auquel la structure est associée. Les domaines auxquels appartiennent les séries ne sont pas des régions spatiales (définies par des coordonnées {x,y,z} de l'espace géographique), ce sont plutôt des régions d'un espace conceptuel de solution.

Tout comme le maillage, la série possède des nœuds et des éléments, lesquels sont utilisés pour interpoler à partir de valeurs portées par les nœuds. L'emplacement de chacun de nœuds est spécifié via une coordonnée.

Typiquement, la coordonnée d'une série est entière ou réelle. Elle pourrait toutefois être un nombre complexe ou encore avoir été définie par l'usager (à condition qu'elle implémente certaines opérations nécessaires).

Sous Modeleur2, les maillages sont de type bi-dimensionnel (2D) avec des coordonnées spatiales {x,y}. Les séries, elles, peuvent être en 1D et 2D, et théoriquement, pourraient même être en n dimensions. Chacun des axes d'une série peut être basé sur n'importe quel type de coordonnées (ex : paramètre de calcul, débit, temps, etc.).

Tout comme les maillages éléments finis, les nœuds d'une série portent des valeurs. Dans le cas des séries, les valeurs peuvent être simples comme des scalaires (ex : niveau d'eau mais peuvent être également des champs répartis spatialement.

Les données d'une série doivent être en relation. Par exemple, une série temporelle doit contenir des données qui diffèrent selon le temps. Il est possible de créer une série temporelle où les valeurs nodales de la série sont des champs de niveaux d'eau ou de vitesses à différents moments de l'année. La région d'une telle série est le domaine du temps. Un champ de niveaux d'eau ou de vitesses peut alors être interpolé pour connaître le niveau d'eau ou la vitesse à tout moment durant la période de temps couverte (région de la série).

L'espace solution que décrit une série peut avoir plusieurs coordonnées, chronologique ou paramétrique. Par exemple, une série de champs de température pourrait être définie en fonction du temps et de la force du vent.

La Figure 12 présente le cas d'une série 1D sur l'axe Q (débit en m3/h). Les coordonnées des nœuds sont des entiers avec les valeurs suivantes : q0 = 200 m3/h, q1 = 400 m3/h, q2 = 600 m3/h.

La région de la série couvre la plage Q = 200 m3/h à Q = 600 m3/h dans l'espace solution. La valeur portée par chacun des noeuds est un champ de niveaux d'eau (variable physique qui dépend du débit). Avec cette série, l'usager est en mesure d'obtenir un champ de niveaux d'eau interpolé pour n'importe quel débit Q dans l'intervalle 200 m3/h à 600 m3/h). L'exemple suivant illustre le champ obtenu à Q = 500 m3/h. L'élément qui englobe Q = 500 m3/h est celui constitué des noeuds avec les coordonnées q1 et q2. Le champ interpolé est donc obtenu à l'aide des champs portés par ces deux nœuds.

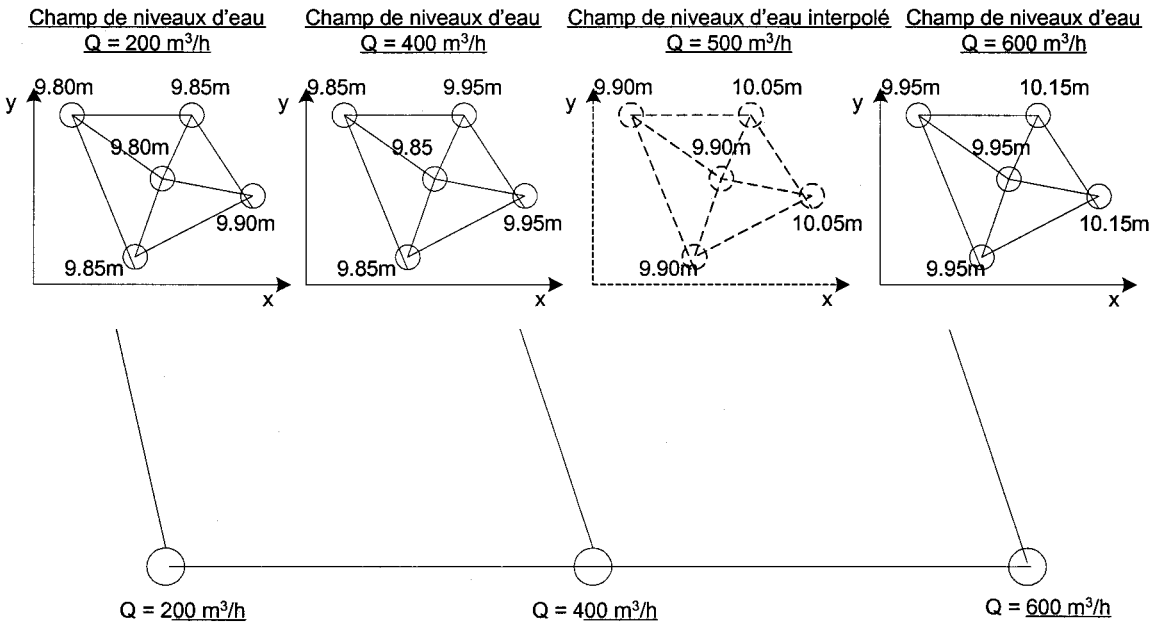


Figure 12 - Exemple de série

Les séries peuvent être combinées entre elles de manière à créer de nouvelles séries. Pour être combinées, deux séries doivent être compatibles. Des séries sont compatibles si et seulement si elles possèdent un axe commun. Par exemple, une série de débits dans le temps est compatible avec une série de températures dans le temps pour créer une série de débits en fonction de la température.

Il est possible de créer une série temporelle où les valeurs nodales sont des champs qui dépendent du débit d'une rivière (ex : les champs de vitesse et de niveaux d'eau), donc d'une série hydrologique. Si le débit peut être interpolé pour en connaître la valeur en un moment particulier, il en est de même des champs de valeurs correspondant au débit.

Dans Modeleur2, l'usager interagit avec les champs de résultats par l'intermédiaire des séries plutôt que directement avec les champs eux-mêmes comme c'était le cas dans Modeleur 1.0. Partout où l'usager est appelé à travailler sur un champ, qu'il soit déjà défini dans l'espace de solution ou non, l'usager le fait via une série (temporelle ou paramétrique) qui caractérise son évolution ou sa variabilité. Cela permet à l'usager de sélectionner un point quelconque dans la série et d'obtenir par interpolation un champ à ce point.

L'avantage de cette approche est de limiter le nombre de calculs à réaliser (en termes statistiques, on parlerait d'échantillon) pour caractériser l'espace de solution (similairement, la population) et d'exploiter à leur maximum les analyses de sensibilité paramétriques, courantes en modélisation.

4.1.3 Développement

Beaucoup de composantes logicielles ont été développées au cours du projet SEGRI. Elles l'ont toujours été dans l'optique de rendre le logiciel Modeleur2 plus flexible et ouvert à des extensions futures.

Dans les prochaines pages, les tâches clé du projet, la motivation derrière la tâche, le ou les objectifs recherchés, les décisions prises de même que les principaux défis rencontrés sont exposés. La plupart des sujets discutés ont fait l'objet d'un ou plusieurs stages.

Pour chacun des sujets traités, des réunions techniques ont été tenues. Suite à ces réunions, des rapports de réunion, de même que des rapports de développement ont été rédigés. Toutes les étapes d'un développement rigoureux ont été suivies (analyse, design, implémentation, test).

Afin de donner une idée de l'envergure de la tâche, la durée du travail est indiquée (unité mois-personne). Dans plusieurs cas, le travail réalisé lors d'un stage a dû être poursuivi par les développeurs senior ou par d'autres stagiaires. Très souvent, le développement d'un module a nécessité d'effectuer des modifications dans d'autres modules ou encore dans d'autres sections de code utilitaire (dont on ne fait pas mention dans cette section).

Principalement en raison de ce fait, il est relativement difficile de quantifier l'effort de travail sur chacun des modules avec précision. Le nombre de mois-homme indiqué pour chacune des tâches se veut plus un indicateur qu'un décompte exact du temps passé sur une tâche précise.

Dans l'estimation des temps, il faut également tenir compte d'un facteur non négligeable: le temps d'adaptation nécessaire à un stagiaire pour s'acclimater et apporter une contribution tangible au projet. Ce temps d'adaptation varie d'un candidat à l'autre, mais se situe généralement dans un horizon de deux semaines, alors que typiquement le stage dure 15 semaines.

Les tâches sont présentées dans un ordre logique qui ne correspond pas nécessairement à l'importance ou la durée de la tâche.

4.1.3.1 Base de données (BD)

Motivation :

Le passage à une base de données relationnelle devait se faire afin de pouvoir gérer des jeux de données de grande taille via des requêtes spatiales (SIG), mais également afin d'ouvrir le logiciel et d'ainsi rendre disponible les données aux utilisateurs via des moyens standards (BD, ODBC, langage SQL).

Objectif :

L'objectif de cette tâche était de concevoir un modèle de données relationnel à partir des structures de données de Modeleur2 (maillage, champs, séries, partitions, etc.) et d'implanter cela dans une base de données relationnelle avec support spatial.

Description :

Initialement, les structures de Modeleur 1.0 ont été extraites et traduites dans un design de base de données relationnel. Par la suite, se sont greffées toutes celles faisant partie de Modeleur2.

Tout au long du projet, le modèle de données fut constamment amélioré et modifié en fonction des développements réalisés et de l'expérience acquise.

Le modèle de données, dans son état actuel, modélise les séries, les champs (analytique/éléments finis, scalaire/vectoriel/ellipse erreur), les maillages, les MNT, les partitions, les couches, les semis de points, les objets graphiques, etc. bref toutes les données disponibles dans Modeleur2.

La plupart des modules ont des composantes dans la BD et il est nécessaire de construire des requêtes SQL pour accéder à l'information dans la ou les tables qui modélisent les données de ce module.

Décision(s) :

- Choix de PostgreSQL (jumelé à PostGIS) comme base de données pour le développement (raisons principales: gratuit et offre un plein support SIG).

- Choix de Case Studio 2 comme outil pour effectuer le design de la base de données (raisons principales : support/optimisation pour multiple BD, faible prix, gestion des versions).

Défi(s) :

- Apprendre les concepts et techniques propres aux bases de données relationnelles.

- Modéliser les structures complexes au cœur de Modeleur 2.0 et garder une façon de faire uniforme pour chacune d'elles.

Durée :

- 5 mois-personne.

4.1.3.2 Module gestionnaire BD

Motivation :

La gestion des données est critique dans un logiciel comme Modeleur. Les données étaient gérées par fichiers dans Modeleur 1.0 alors que dans Modeleur2, elles le sont via une base de données relationnelle.

Il était requis de développer un système qui allait permettre de centraliser les opérations avec la base de données afin de garantir intégrité et robustesse, mais en étant assez distribué pour s'intégrer dans une architecture avec plug-in.

Objectif :

L'objectif de cette tâche était de mettre en place un module central qui allait gérer les interactions avec la base de données mais sans contenir pour autant le code des algorithmes BD pour chaque structure de données (si c'eut été le cas, il aurait fallu rouvrir ce module à chaque fois qu'on ajoute une nouvelle structure).

Description :

Ce module est responsable de la gestion des projets en tant que container de données. Il implante toutes les opérations liées au projet et a la responsabilité de lister les connexions ODBC.

Pour sa part, le projet est responsable de rediriger les opérations (charge, sauve, efface etc.) demandées par les autres modules vers la classe d'algorithme BD appropriée.

Le module assure la cohérence et l'intégrité des données en utilisant les mécanismes de transaction. Il gère le compte de référence des données en mémoire, en plus d'être le responsable des métadonnées, des accès et des types.

Décision(s) :

- Utiliser une mécanique d'algorithme BD (GDAlgoIo) dans laquelle un algorithme est responsable des opérations suivantes : chargement / déchargement en mémoire et en BD, sauvegarde / sauvegarde sous, effacement mémoire et en BD, exportation / importation.

- Les algorithmes BD peuvent être utilisés en cascade, mais ceci doit se faire via le projet, car lui seul maintient le compte des références.

Défi(s) :

- Mettre en place les mécanismes pour que les plug-in puisse inscrire des algorithmes BD (GDAlgoIo).

- Assurer l'intégrité totale et complète des données de Modeleur2, en tout temps.

- Construire des requêtes SQL relativement complexes, lors de l'écriture des algorithmes BD dans les différents modules.

- Optimiser les longs traitements (chargement/sauvegarde) au moyen de certains mécanismes : indexation, paramètres BD, vues, requête SQL optimale, langage pg/plsql.

Durée :

- 10 mois- personne.

4.1.3.3 Module scripts & macros

Motivation :

L'interface graphique de Modeleur 1.0 était développée avec une librairie qui n'était plus supportée (non disponible à des tiers).

Le langage de la calculatrice étant un langage propriétaire, il ne pouvait pas être facilement étendu sans nécessiter des efforts de développement important.

Modeleur 1.0, de la façon dont il est construit, n'offrait pas d'ouverture pour effectuer des ajouts ou des améliorations au logiciel.

Objectif :

Le principal objectif de cette tâche était de déterminer un langage de script pour prendre en charge l'interface graphique du logiciel, remplacer le langage de la calculatrice et permettre la mise en place d'une architecture avec plug-in (ouverture du logiciel).

Le second objectif était de trouver toute une gamme d'outils pour développer efficacement avec ce langage et surtout permettre l'interopérabilité avec le C++ pour qu'il soit possible de rendre disponibles les structures de données du logiciel à des scripts.

Le troisième objectif était de choisir un langage qui allait servir à l'écriture de scripts de test.

Le quatrième objectif, moins critique toutefois, était d'implanter à l'aide du langage trouvé une fonctionnalité permettant d'enregistrer et de rejouer des macros.

Description :

Après avoir étudié en détails ce qui était disponible sur le marché, Python/wxPython fut choisi comme langage de script pour Modeleur2. Il était le seul à répondre aux critères de gratuité, d'interopérabilité avec le C++ et de flexibilité nécessaires à la mise en place du système.

Le choix de Python comme langage fut une étape cruciale qui allait dicter la voie du développement pour toute la durée du projet. Ce langage répond pleinement aux besoins qui étaient exprimés et est au cœur même de toute l'ouverture offerte par le logiciel.

Décision(s) :

- Choix de Python/wxPython comme langage de script (raisons principales: gratuit, interopérable avec le C++, flexible).

- Choix des outils intégrés à Modeleur2 : Boa Constructor comme IDE et Winpdb comme débogueur.

- Utilisation de Boost-Python pour rendre disponible les structures du langage C++ dans le langage Python.

Défi(s) :

- Apprendre le langage Python et développer une façon d'écrire du code (normes de programmation).

- Apprendre à publier en Python avec Boost-Python et Pyste, et maîtriser les subtilités reliées à la publication en Python de structures C++.

- Utiliser multiples instances de l'interpréteur Python (nécessaire à l'exécution de scripts).

- Etre en mesure d'extraire le contenu d'un événement qui circule sur le bus, et de sauver celui-ci dans un fichier (requis pour la fonctionnalité des macros).

- Intégrer un débogueur externe pour déboguer le code de scripts Python en ayant accès à l'espace d'exécution de Modeleur2.

Durée :

- 5 mois- personne.

4.1.3.4 Plug-in & bus logiciel

Motivation :

Sous Modeleur 1.0, il n'était pas possible d'effectuer des ajouts au logiciel. Lorsqu'une fonctionnalité était requise, tout le logiciel devait être reconstruit en entier, processus long et laborieux. Modeleur2 se devait d'être plus ouvert et de permettre d'y ajouter des composantes externes une fois une version officielle livrée.

Objectif :

L'objectif de cette tâche était de concevoir un système auquel on peut ajouter des composantes au démarrage au moyen d'un script d'initialisation.

Description :

Le système est centré autour d'un bus logiciel auxquels se connectent des modules qui interagissent entre eux au moyen d'événements. Chaque module est composé d'un gestionnaire qui est responsable de traiter les événements qui l'intéresse.

Les plug-in peuvent être constitués d'une partie C++ (encapsulée dans une DLL) ou d'une partie Python. Ils peuvent ajouter des composantes dans la zone de contrôle, ajouter des menus et des barres d'outils. Ils ont la responsabilité d'ajouter les algorithmes BD nécessaires à leur fonctionnement ou à celui d'autres modules.

Ce système de plug-in constitue une grande innovation de Modeleur2 car il permet d'ouvrir le logiciel aux contributions externes.

Décision(s) :

- Utiliser un bus d'événements comme seule plateforme de communication entre les modules (raison principale : minimiser le couplage entre les modules; pré requis à l'architecture avec plug-in).
- Construire le système de plug-in autour du langage Python.

Défi(s) :

- Implanter une mécanique simple pour l'ajout de plug-in (développeurs externes).
- Permettre l'ajout de plug-in C++, Python, ou hybride C++/Python.
- Permettre aux plug-in d'ajouter des algorithmes BD.
- Implanter un bus logiciel pour la distribution des événements (callback).

Durée :

- 5 mois- personne.

4.1.3.5 Interface graphique & boîtes de dialogue

Motivation :

L'interface graphique de Modeleur 1.0 n'était pas standard et intuitive pour les nouveaux usagers. La façon de fonctionner avec des modes exclusifs gérés par des menus devait être changée pour quelque chose de plus souple, de plus intuitif pour l'usager.

Objectif :

L'objectif était de concevoir une interface graphique et des composantes GUI qui allaient être intuitive pour l'usager tout en correspondant aux besoins des différents modules du système.

Description :

Le travail a débuté sur à l'aide de réunions et des croquis ont été réalisés sous Visio. Un prototype a été développé et différents ajouts s'y sont greffés.

Plusieurs classes Python ont été conçues pour gérer les différents aspects de l'interface graphique : menus, barre d'outils, zone de contrôle.

Des boîtes de dialogue ont été développées pour répondre aux besoins du logiciel : choix d'entité, gestion d'entité, gestion de la zone d'étude, etc.

L'accent a été mis sur le développement de mécanismes ou de classes qui allaient pouvoir être réutilisés par les autres modules plutôt que sur l'esthétisme du logiciel en soit.

Décision(s) :

- Choix de wxPython comme langage pour développer l'interface utilisateur.
- Choix d'une interface MDI (Multiple Document Interface).

Défi(s) :

- Apprendre wxPython.

Durée :

- 5 mois- personne.

4.1.3.6 Module affichage/visualisation

Motivation :

Modeleur 1.0 offrait une gamme de fonctionnalités de visualisation toute implantées à même le logiciel (beaucoup de code a été produit pour gérer ces aspects et devait être maintenu).

Modeleur2 se devait d'offrir plus de fonctionnalités, notamment l'affichage 3D et un mode interactif pour le déplacement et le zoom, mais en tentant de réduire considérablement le code propriétaire au logiciel.

Objectif :

L'objectif de cette tâche était de fournir aux usagers toute la puissance d'un moteur graphique externe tout en retirant de Modeleur2 le poids du développement et de la maintenance de code traitant des aspects affichage/visualisation.

Description :

Après avoir étudié en détails ce qui était disponible sur le marché, VTK fut choisi comme moteur graphique. Il était le seul à répondre aux critères de gratuité, d'indépendance de plate-forme et de fenêtrage et d'offrir un support 2D/3D.

Le module d'affichage fut donc implanté en se basant sur VTK. Le module a la responsabilité d'afficher les « objets graphiques » qu'il reçoit des autres modules du système. Le module prend également en charge toute la partie zoom et navigation qui est grandement améliorée en comparaison à Modeleur2.

Les fonctionnalités de visualisation (tracé de points, d'isolignes, d'isosurfaces, etc), quant à elles, sont désormais prises en compte directement par VTK par l'intermédiaire de MayaVi. Grâce à MayaVi qui vient se greffer dans la zone de contrôle de Modeleur2, toute la puissance de VTK est offerte aux utilisateurs.

Décision(s) :

- VTK fut choisie comme librairie graphique pour répondre aux besoin de visualisation et remplacer ce qui existait sous Modeleur2.
- Inversion du traitement : plutôt que de demander un tracé, l'usager charge un champ, et ensuite, demande un traitement sur ce champ (ex : affichage isolignes).
- MayaVi fut choisie comme composant externe pour la visualisation.

Défi(s) :

- Apprendre VTK;
- Élaborer un système autour de cette librairie (VTK).
- Intégrer MayaVi dans Modeleur2.

Durée :

- 6 mois- personne.

4.1.3.7 Module MNT

Motivation :

La construction d'une partition dans Modeleur 1.0 se faisait en juxtaposant des sous domaines. Quand le nombre de sous domaines devenait élevé, les contrôles de cohérences géométriques rendaient toute opération extrêmement lourde.

Objectif :

L'objectif de cette tâche était d'implanter le MNT comme un container de partitions conçu sur un tout nouveau modèle de partitions basées sur une superposition de couches pour éviter d'avoir à effectuer des contrôles géométriques, et retomber dans le même problème que Modeleur 1.0.

Description :

Les grands acteurs ont été identifiés et les scénarios d'usage ont été extraits pour bien établir les échanges entre le GUI, le gestionnaire C++, l'éditeur, l'affichage et le gestionnaire de données.

Les classes ont été construites pour pouvoir être réutilisées dans le module partition de maillage, tout comme la composante arborescence utilisée pour gérer les MNT ouverts.

Sous Modeleur2, les couches sont liées à un jeu de données, et la limite de ce jeu de données doit être respectée. Cette vérification est faite à la sortie de l'édition.

Le module MNT s'occupe de l'assemblage des partitions sur un maillage, en fonction de la couche la plus prioritaire.

Décision(s) :

- Développer un module Éditeur indépendant qui sera utilisé pour l'édition des couches, mais aussi partout ailleurs dans le logiciel.

Défi(s) :

- Garder les classes du MNT intactes (sans objets propres à Modeleur2) pour qu'il soit possible d'utiliser le groupe de classes dans un contexte autre que Modeleur2.
- Établir la façon de faire les échanges d'informations entre les différents intervenants (design de référence lors de la conception des modules subséquents).
- Concevoir des classes réutilisables dans le module des partitions de maillage.

Durée :

- 6 mois- personne.

4.1.3.8 Module éditeur géométrique

Motivation :

L'éditeur géométrique de Modeleur 1.0 remplissait les besoins. Celui de Modeleur2 doit effectuer les mêmes tâches, mais être basé sur le moteur graphique qui fut choisi (VTK).

Objectif :

L'objectif de cette tâche était d'implanter un éditeur géométrique générique (basé sur VTK) qui allait pouvoir être utilisé par n'importe quel autre module qui allait avoir des besoins d'édition dans Modeleur2.

Description :

L'éditeur permet l'édition de multipoints, de multipolylignes et de multipolygones. Il supporte les actions de sélection, suppression, de déplacement, d'union et d'insertion sur les objets appropriés. L'éditeur utilise les fonctionnalités de VTK pour effectuer son travail.

Les modules ayant des besoins d'édition font parvenir au module éditeur l'objet à éditer au moyen d'un événement. L'objet en question doit hériter d'une interface offerte par l'éditeur.

C'est la structure en édition qui doit permettre ou refuser les modifications (ex : déplacement d'un point) que l'éditeur demande car lui-même n'est pas au courant des contraintes qui peuvent exister sur une structure.

Lors du développement de ce module, un gros effort a été mis sur les géométries et sur les géométries porteuses de valeurs (champ de données vectorielles au sens SIG).

Décision(s) :

- L'éditeur propose une interface dont doivent hériter les structures éditables (ex : les couches d'une partition, les points d'un semi).

Défi(s) :

- Apprendre VTK.

- Implanter une interface basée sur des itérateurs sur les structures géométriques pour simplifier l'utilisation de l'éditeur et la compréhension du code.

- Implanter des classes de géométries Modeleur2 qui répondent aux besoins de tout le logiciel.

Durée :

- 5 mois- personne.

4.1.3.9 Module partition de maillage

Motivation :

Modeleur 1.0 permet d'effectuer l'opération de maillage en spécifiant une taille de maille. Modeleur2 devait implanter un mécanisme de maillage adaptatif où l'opération de maillage se fait plutôt à partir de champ d'erreur et d'un niveau de tolérance à l'erreur.

La construction de partition de maillage souffrait également du même problème que le MNT (lourdeur des contrôles géométriques).

Objectif :

Le principal objectif de cette tâche était d'implanter un module qui allait permettre d'utiliser la mécanique des partitions avec une superposition de couches configurées avec des paramètres nécessaires à l'opération d'un mailleur.

L'objectif secondaire était de développer le module pour qu'il soit indépendant du mailleur utilisé, et ainsi laisser la porte ouverte à l'ajout de mailleurs externes.

Description :

Un système « par proxy » a été développé pour rendre transparente l'utilisation d'un mailleur ou d'un autre. Une interface de mailleur 1D et 2D a été développée et les mailleurs doivent en hériter pour pouvoir être inséré dans Modeleur2. Quelques mailleurs ont d'ailleurs été ajoutés à Modeleur2 à l'aide de cette façon de faire: mailleur Frontal, mailleur Delaunay et mailleur Trivial.

Lors du développement, un important travail a été réalisé pour développer un algorithme efficace qui traite tous les cas de figure au niveau géométrique.

Tel que prévu, beaucoup d'éléments du module MNT ont pu être réutilisés. Il fut tout de même nécessaire d'effectuer des ajouts et de synchroniser les deux modules à la fin du développement.

Décision(s) :

- Utiliser la librairie GEOS pour effectuer les fonctionnalités géométriques au lieu de Terralib car elle ne répondait plus aux besoins.

- Utiliser un système « par proxy » pour supporter l'ajout de mailleurs; technique validée et utilisée ailleurs dans le logiciel.

Défi(s) :

- Développer un algorithme de maillage efficace et fonctionnel qui utilise les paramètres de la couche la plus prioritaire, mais qui garde la continuité aux frontières.

- Migrer les mailleurs de Modeleur 1.0 sous Modeleur2 et démontrer qu'il sera aisé d'intégrer des mailleurs externes.

Durée :

- 10 mois- personne.

4.1.3.10 Module de validation & filtrage

Motivation :

Avec l'avènement des campagnes de terrain par laser aéroporté, la taille des jeux de données ne cesse d'augmenter. L'ouverture et l'affichage de semi de points de grande taille sous Modeleur 1.0 n'était pas possible car toutes les données devaient être chargées en mémoire.

De plus, le logiciel n'offrait aucun moyen de valider les données autre que manuellement; ce qui est impensable quand il est question de millions voire de dizaine millions de points.

Objectif :

Le premier objectif était de concevoir un système de tuiles pour éviter d'avoir à charger tous les points d'un semi de points.

Le second objectif était de concevoir un module qui allait être en mesure d'accueillir des filtres de validation de points développés par les usager soit en C++ (via une DLL) ou en Python.

Description :

Un tuile est un artifice qui permet de séparer un semi de points en sections qui englobent un certains nombre de points. Le système en place permet de ne charger que les tuiles qui sont visibles dans la zone d'affichage plutôt que de tout charger. Si l'usager se déplace dans son semi, les tuiles qui ne sont plus visibles seront déchargées et de nouvelles seront chargées. Si l'usager modifie des points, les modifications sont emmagasinées de façon

temporaire dans la base de données jusqu'à ce que l'usager confirme qu'il souhaite conserver ses changements par une opération de sauvegarde.

Le module validation intègre la fenêtre de propriétés des points et l'arborescence dans la zone de contrôle. La fenêtre de propriétés dépend du type de semi de points.

La notion de zone d'étude a été mise en place et peut être utilisée partout dans le logiciel pour délimiter la portée spatiale d'une requête SQL.

Décision(s) :

- Utiliser la base de données pour le stockage temporaire des points modifiés.
- Utiliser le XML pour décrire les filtres primaires et secondaires.
- Utiliser un système « par proxy » pour les filtres.

Défi(s) :

- Implanter une façon de faire différente des autres modules en ce qui a trait aux algorithmes BD (dû au fait que la BD est utilisé pour le stockage temporaire).
- Développer le module de façon à pouvoir accueillir tout type de semi de points dans l'avenir.
- Publier et utiliser l'interface C+ pour réaliser un filtre Python
- Rendre transparente l'utilisation de filtres C++ ou Python.

Durée :

- 6 mois- personne.

4.1.3.11 Maillage, champs et séries

Motivation :

Modeleur 1.0 permettait de réaliser des calculs sur des champs via la calculatrice. Modeleur2 vise entre autres à permettre d'effectuer des calculs sur des champs interpolés à partir d'une série.

Objectif :

L'objectif de cette tâche était de mettre en place dans Modeleur2 les structures fondamentales du logiciel (les maillages, les champs et les séries) et de rendre certaines d'elles disponibles en Python.

Description :

La mise en place du concept de série a affecté les autres structures de données du logiciel, notamment les maillages et les champs. Il n'est pas trivial d'articuler un concept aussi abstrait et complexe en langage informatique. L'implantation des séries a donc nécessité un important effort qui s'est poursuivi tout au long du développement du logiciel.

Les classes ont sans cesse été améliorées, polies et restructurées pour répondre aux besoins du logiciel. Une partie du code provenant de Modeleur 1.0 a pu être réutilisée, principalement celui traitant de l'aspect calcul numérique.

Les maillages sont désormais une classe générique qui sont disponibles en 0D, 1D et 2D. Les champs peuvent être soit éléments finis ou analytiques et porter des valeurs de type scalaire, vectorielle ou ellipse erreur. Il est maintenant possible de construire et de gérer des séries ayant comme valeur portée des réels ou encore des champs, en plus de pouvoir interpoler sur ces séries.

Le concept de région (0D, 1D, 2D) a été mis en place, de même que celui d'itérateur. Grâce à ces deux innovations, il est possible d'itérer sur un champ ou une série en spécifiant un pas.

Décision(s) :

- Implanter les classes maillage, champs et série à l'aide de « template ».

Défi(s) :

- Pour les stagiaires, arriver à saisir conceptuellement les notions abstraites de séries, de champs, de régions, d'itérateurs et d'arriver à exprimer cela en informatique.

- Développer une hiérarchie complexe de classes « template » et traduire cette hiérarchie de classes en Python.

Durée :

- 10 mois- personne.

4.1.3.12 Éditeur de séries & éditeur de champs analytiques

Motivation :

Les nouvelles structures de données des séries et des champs ellipse erreur devait pouvoir être construites et gérées par les usagers.

Objectif :

Le premier objectif était de concevoir un éditeur de séries qui allait permettre de construire et de gérer des séries.

Le second objectif était de concevoir un éditeur de champs analytiques pour construire les champs ellipse erreur nécessaires aux couches de maillage.

Description :

L'éditeur de série permet à l'usager de construire et de gérer des séries 1D ou 2D qui portent soit des réels, soit des champs. L'usager peut spécifier les coordonnées de nœuds de sa série, ou les faire générer automatiquement en spécifiant un pas et un nombre de nœuds.

L'éditeur de champs analytiques permet de construire et gérer des champs analytiques. L'usager peut écrire une équation analytique et demander à l'interpréteur de valider la syntaxe. Les champs analytiques ellipse erreur sont requis par les couches de maillage.

Décision(s) :

- Dans un premier temps, les séries sont constituées uniquement d'éléments Q4.

Défi(s) :

- Implanter l'utilisation d'un interpréteur à part pour évaluer les fonctions des champs analytiques.

- Retourner les valeurs portées par les séries, par valeur (ex : réel) ou par référence (ex : champ).

Durée :

- 4 mois- personne.

4.1.3.13 Module de simulation

Motivation :

Le seul simulateur qui est supporté dans Modeleur 1.0 est Hydrosim. L'ajout d'un simulateur ne pouvait se faire qu'en ouvrant le logiciel pour y intégrer le simulateur.

Le nouveau simulateur développé pour Modeleur2 est H2D2 et il permet de réaliser différents types de simulation (ex : simulation de température, simulation hydrodynamique, etc.).

Objectif :

L'objectif de cette tâche était de concevoir un module capable d'accueillir un ensemble simulateur constitué de l'exécutable du simulateur et des composantes nécessaires à son fonctionnement.

Description :

Le module simulation est responsable de gérer les simulations en tant que container de calculs. Un ensemble simulateur fournit quant à lui une classe de calcul qui est propre à son simulateur (exécutable).

La classe de calcul reçoit des objets import/export qui regroupe des informations décrivant les entités à exporter/importer, les fichiers source/destination de même que les filtres nécessaires à l'opération d'import/export.

Décision(s) :

- Le design est fait avec un modèle d'ancrage par héritage (la classe calcul spécialisée hérite de celle fournit par le module simulation).
- Un ensemble simulateur doit contenir: les filtres (importateurs et exportateurs), la partie GUI du simulateur, la DLL du simulateur fournissant les algorithmes BD spécialisés et les classes spécialisées notamment la classe responsable de l'exécution du calcul.
- Le module fait appel à une fonctionnalité d'import/export pour faire connaître les fichiers au simulateur et rapatrier les résultats fournis par celui-ci.

Défi(s) :

- Exprimer les besoins des simulateurs et concevoir un module générique.

Durée :

- 4 mois- personne.

4.1.3.14 Import/export

Motivation :

Certains utilisateurs ont fait savoir qu'ils trouvaient les opérations d'import/export longues et fastidieuses, principalement parce qu'il n'était pas possible d'effectuer plusieurs opérations d'import/export en une seule commande.

Objectif :

L'objectif de cette tâche était d'implanter un module capable de bien gérer l'importation et l'exportation des données Modeleur2 tout en s'intégrant à l'architecture développée.

Description :

Une boîte de dialogue permettant d'effectuer l'import/export a été développée. Elle permet d'effectuer une opération d'import/export sur plusieurs entités. La fonctionnalité d'import/export fait appel au concept de filtre qui traduit les données dans le format intermédiaire connu de Modeleur2.

Il est possible d'importer/exporter les structures suivantes : maillages, champs éléments finis scalaire et vectoriel et semi de points de topographie.

Il est également possible d'importer/exporter les structures de Modeleur2 vers le format MapInfo et d'importer des structures du format DXF.

Décision(s) :

- Traduire les fichiers dans un format intermédiaire comme ce fut fait dans Modeleur 1.0.

Défi(s) :

- Intégrer un système d'entête avec format XML à tous les fichiers.
- Intégrer le système d'import / export aux algorithmes BD actuels.
- Développer une interface graphique qui permette d'effectuer l'import/export de plusieurs fichiers simultanément, sans fermer la fenêtre.

Durée :

- 3 mois- personne.

4.1.3.15 Traducteur

Motivation :

Il est nécessaire de pouvoir utiliser les données des anciens projets de Modeleur 1.0.

Objectif :

L'objectif de cette tâche était de concevoir un outil pour effectuer la traduction des projets de Modeleur 1.0 dans le modèle de données de Modeleur2.

Description :

Le traducteur a été conçu dans l'environnement Borland 5.1 pour pouvoir réutiliser les librairies de Modeleur 1.0 sans avoir à reconstruire tout le code de Modeleur 1.0.

Tout au long du projet, le traducteur fut revisité pour y effectuer les modifications impliquées par les divers changements faits dans la base de données.

Le traducteur de données est en mesure de traduire les maillages, les champs vectoriels, les champs scalaires de même que les semis de points de topographie.

Décision(s) :

- Réutiliser les librairies de Modeleur 1.0 (donc développer sous Borland C++ 5)

Défi(s) :

- Travailler dans l'environnement de développement Borland C++ 5, environnement auquel peu de développeurs étaient familiers.

- Travailler avec le langage C++ (1995) avant sa standardisation (certaines fonctionnalités de la librairie standard ne sont donc pas disponibles aux développeurs).

Durée :

- 5 mois- personne.

4.1.3.16 Banc de test

Motivation :

Les tests sont une étape critique de la validation d'un logiciel. Modeleur2 se devait évidemment d'être testé.

Objectif :

L'objectif de cette tâche était de concevoir et mettre en place un banc de test qui allait permettre d'effectuer des tests de façon automatisée et de publier les résultats des tests sur une page web.

Description :

Le banc de test est constitué d'un ensemble de scénarios de test qui ont pour objectif de valider le comportement du logiciel et des plug-in qui le composent. Les tests sont faits en envoyant sur le bus des événements qui correspondent aux fonctionnalités à tester. Des scripts Python génériques ont été conçus et sont réutilisés dans tous les scripts (ex : chargement, sauvegarde, etc.).

L'exécution d'un script de test amène la création d'un fichier de log qui indique si le test est un succès ou un échec. Dans le cas d'un échec, le message d'erreur est inclus dans le log. Ce fichier de log est ensuite lu par un post-traitement qui vise à publier les résultats sur le site web du groupe. Les scripts de tests permettent de valider les cas corrects d'utilisation d'une fonctionnalité, mais ils permettent également de vérifier si le bon code d'erreur est envoyé lors d'une utilisation incorrecte.

Un projet type a été construit, lequel contient tout les éléments nécessaires pour l'exécution des tests. La base de données contenant ce projet a été enregistrée dans un fichier, et à chaque exécution d'un scénario de test, cette base de données « pure » est effacée et restaurée. On s'assure ainsi d'avoir toujours le même environnement lors des tests.

La structure du banc de test a été validée et s'avère fiable et robuste. Au moment d'écrire ces lignes, le banc de test contient un total de 185 scénarios de test. La banque de scénarios de test continuera de prendre de l'ampleur dans les prochains mois ou dans les prochaines années à mesure que le développement se poursuivra.

Décision(s) :

- Les tests se font par envoi d'événements sur le bus.

Défi(s) :

- Concevoir des scénarios de tests complets, documentés et faciles à maintenir.

- Profiter des tests pour réviser et clarifier l'interface des événements car c'est via cette interface que les usagers pourront utiliser les fonctionnalités du logiciel via des scripts.

Durée :

- 5 mois-personne.

4.1.3.17 Système de validation automatisé

Motivation :

Le code produit doit respecter le standard C++ et doit être vérifié quotidiennement pour s'assurer de détecter les erreurs dans la banque de code commune à tous les développeurs.

Objectif :

L'objectif de cette tâche était de concevoir un système de compilation multi-compilateur automatisé et de publier les résultats sur une page web.

Description :

Le système inclut un logiciel qui analyse le contenu des fichiers de solution Visual Studio et génère un fichier utilisable par l'outil Boost.Jam qui exprime les dépendances de chaque module.

Des fichiers de configuration ont été construits pour chaque compilateur et sont utilisés par le système automatisé, mais également par les développeurs, ce qui fait que les paramètres utilisés sont les mêmes dans les deux cas.

Décision(s) :

- Le choix de Boost.Jam comme outil pour effectuer la compilation automatisée (raison principales : gestion des dépendances, gestion de multiples compilateurs).

Défi(s) :

- S'assurer que le code construit soit compatible avec tous les compilateurs que l'on désire supporter (VC 7.0, VC 7.1, Intel C++ 9.0, gcc 3.3, en mode « debug » et « release »).

- Monter un système complet qui extrait quotidiennement les sources de CVS, qui recompile les fichiers qui ont changé, puis reconstruit les modules en fonction de leurs dépendances et qui publie les résultats sur un site web.

Durée :

- 5 mois-personne.

4.1.4 État du logiciel

Avant d'effectuer un survol des fonctionnalités disponibles dans la version livrée, il apparaît pertinent de jeter un coup d'œil à l'état du logiciel dans son ensemble.

Le logiciel offre toute la souplesse et la flexibilité pour être étendu dans l'avenir. Les fonctionnalités d'intégration de composantes (filtres, mailleurs, simulateurs), la publication des structures fondamentales (maillage, champs, séries) dans les scripts et la mécanique des plug-in sont autant de faits qui illustrent toute cette souplesse, cette ouverture vers l'extérieur présente dans Modeleur2.

L'implantation des « séries » en tant que structure de données ouvre la porte à toute une gamme de calculs. Cette facette de Modeleur2 a nécessité un effort de travail considérable car beaucoup d'aspects ont dû être revus pour l'intégrer au reste. Les séries peuvent être construites et manipulées grâce à un éditeur de séries inclus dans le logiciel. Leur utilisation permet d'envisager toute une gamme d'applications qui étaient autrefois impossibles.

L'implantation du modèle des partitions avec couches superposées a fait ses preuves. La construction de couches est désormais un exercice convivial qui ne demande pas de réaliser le travail minutieux et parfois lourd de juxtaposer des sous-domaines comme c'était le cas dans Modeleur 1.0.

Le fait de faire appel à un système externe et spécialisé pour les tâches de visualisation offre une puissance inégalée en ce qui a trait aux capacités d'analyse offertes. L'intégration de VTK et de MayaVi permettent à Modeleur2 de mettre l'accent sur ses fonctionnalités propres. Les fonctionnalités de visualisation offertes par MayaVi sont disponibles dans un onglet de la zone de contrôle de Modeleur2.

Le fait de travailler avec une base de données relationnelle donne accès à toute une gamme de services qui assurent l'intégrité des données et le contrôle des accès. Elle libère toute une puissance en ce qui a trait aux requêtes qui peuvent être faites sur les données. La fonctionnalité SIG permet pour sa part de tirer profit de requêtes SIG. Une architecture fonctionnelle est en place et les interactions avec la base de données sont bien gérées.

Malgré tous les avantages qu'elle apporte, la mise en place de la base de données (combinée à l'inexpérience du groupe dans ce domaine) a amené son lot de problèmes. En effet, dans l'état actuel, les performances lors du chargement données sont très en deçà des attentes. Le chargement/sauvegarde des maillages, des champs ou des semis prend un temps considérable.

Priorisant la mise en place de fonctionnalités clés, peu de temps a pu être investi pour trouver des moyen d'améliorer les performances de la base de données. C'est un aspect qui devra être révisé pour qu'il soit possible d'utiliser Modeleur2 avec des cas d'utilisation réel comme la rivière Châteauguay, par exemple. Diverses hypothèses pouvant expliquer la faible performance ont été émises (indexation, requêtes SQL non optimales, paramètres de base de données inadéquats, librairie OTL/ODBC, etc.). Des investigations et des tests seront requis pour corriger cette lacune.

Dans le même ordre d'idée, peu de la force de travail a été concentrée sur l'aspect esthétique, avec résultat qu'à la fin du projet, l'interface usager n'est pas aussi relevée qu'elle pourrait l'être.

Enfin, il apparaît important de mentionner que par manque de temps et de ressources, certains aspects du logiciel n'ont pu être abordés. Les partitions de conditions aux limites et la publication des résultats en sont un exemple.

De plus, certaines fonctionnalités sont incomplètes, comme c'est le cas des activités de validation des données et de construction du MNT qui ne supportent que les données de topographie. Le module de simulation, pour sa part, bien qu'il soit relativement avancé, n'a pu être finalisé et il n'est donc pas possible d'exécuter des simulations avec la version livrée.

Enfin, les outils d'analyse suivants tous mentionnés dans les spécifications fonctionnelles n'ont pu être implantés :

- Outil de lignes;
- Outil de surfaces;
- Outil de vue tabulée;
- Outil de projection (projeter un champ lié à un maillage vers un autre maillage);
- Outil de projection cartographique.

Lors des prochaines pages, les fonctionnalités décrites dans les spécifications fonctionnelles seront visitées une à une. Celles qui sont supportées seront mentionnées, mais celles qui s'avèrent incomplètes et non disponibles le seront également afin de donner une image juste de l'état du logiciel livré.

4.1.4.1 Activité : projet et gestion des données

Supporté

- Créer, ouvrir, fermer, sauver et sauver sous des projets.
- Modifier les paramètres d'un projet (zone affichage, projection SIG, zone étude).
- Extraire les connexions ODBC pour l'ouverture/création de projets.
- Créer et de gérer des zones d'étude.
- Créer, ouvrir, fermer, sauver et sauver sous des champs analytiques.

- Voir les entités du projet (nom, type, id, description).
- Modifier le nom et/ou la description d'une entité.
- Supprimer une entité (avec vérification des dépendances).
- Importer des fichiers, exporter vers des fichiers.
- Import/export: semis de points de topographie, maillages, les champs éléments finis scalaires et vectoriels.
- Importer en format DXF ou MapInfo.
- Importer/exporter en format MapInfo.
- Importer/exporter plus d'une entité à la fois.

Incomplet/non supporté

- Ouvrir et gérer plus d'un projet simultanément.
- Créer des liens externes ou des liens vers d'autres projets.
- Trier les données par nom, par type, etc.
- Raffiner la liste des données à l'aide de critères SQL.
- Consulter les dépendances d'une donnée.
- Importer des données d'une base données, en exporter vers des bases de données.
- Import/export d'autres types de données que ceux mentionnés.
- Le standard FGDC n'est pas supporté.
- La sauvegarde sous d'un projet valide seulement sous PostgreSQL.

4.1.4.2 Activité : validation de données

Supporté

- Validation des données de topographie seulement.
- Créer, ouvrir, fermer sauver et sauver sous un semi de points.
- Ajouter des points et supprimer des points manuellement.
- Spécifier une zone d'étude.
- Sélectionner des points, individuellement ou par zone.
- Activer ou désactiver des points.
- Consulter, modifier les attributs de points.
- Garder la valeur originale des points mesurés.
- Publier un semi de points en champ.
- Filtrer un semi de points (et désactiver des points via le filtre).

Incomplet/non supporté

- Validation des données autres que topographie (ex : substrat).
- Pouvoir rétablir la valeur originale.
- Interdire le déplacement ou la suppression de points mesurés.
- Calculer automatiquement une échelle de précision.
- Combiner des semis de points.

- Gérer des semis de points avec pas de temps.
- Copier, coller des points d'un semi à l'autre.
- Ajouter des points via des scripts.

4.1.4.3 Activité : MNT

Supporté

- Construction de partitions de topographie seulement.
- Gérer les MNT, les partitions et les couches à partir d'une arborescence.
- Créer, ouvrir, fermer, sauver et sauver sous un MNT.
- Ajouter une nouvelle partition, ajouter une nouvelle couche.
- Retirer une partition.
- Éditer la géométrie d'une couche.
- Valider l'édition à la sortie (limite de validité du jeu de données).
- Modifier la priorité d'une couche.

Incomplet/non supporté

- Construction de partitions autres que topographie (ex :substrat)
- Construction de partitions génériques.
- Ajouter une partition existante, ajouter une couche existante
- Copier/coller une partition ou une couche.
- Modifier la couleur d'une couche.
- Annexer des commentaires.
- Générer le squelette du MNT.

Présentement affichée d'un couleur très pâle, la limite de validité d'une couche doit être améliorée. Lors de l'édition des couches, certains déplacements de points provoquent l'affichage d'un voile indésirable. L'éditeur devrait également empêcher d'effectuer certaines actions qui permettent de créer des géométries invalides.

4.1.4.4 Activité : Partition de maillage

Supporté

- Gérer les partitions de maillage et les couches à partir d'une arborescence.
- Créer, ouvrir, fermer, sauver et sauver sous une partition de maillage.
- Ajouter une nouvelle couche.
- Retirer une couche.
- Éditer la géométrie d'une couche.
- Modifier la priorité d'une couche.
- Assigner un champ d'ellipse erreur à une couche.
- Mailler une partition ou une couche.

- Afficher maillage, sauver le maillage.

Incomplet/non supporté

- Spécifier la taille des mailles directement.
- Mailler en fonction du critère de tolérance à l'erreur (maillage adaptatif).
- Mailler les zones avec des trous.
- Copier/coller une couche.
- Modifier la couleur d'une couche.
- Annexer des commentaires.
- Afficher le squelette du MNT (zone où il y a des données).
- Trouver le niveau de tolérance à l'erreur optimal (histogramme de répartition de l'erreur).
- Obtenir de l'information sur le maillage généré.

Les mêmes commentaires concernant l'édition des couches s'appliquent au module partition de maillage.

4.1.4.5 Activité : assemblage des données sur un maillage

Supporté

- Assembler les données sur un maillage.
- Assembler les données provenant de plusieurs partitions.

Incomplet/non supporté

- Permettre l'assemblage par itérations sans fermer la boîte assemblage.
- Fournir de l'information pertinente lors d'un échec de l'assemblage.

4.1.4.6 Outil d'analyse : affichage

Supporté

- Afficher des objets dans les fenêtres graphiques.
- Entrer/sortir d'un mode de zoom interactif (avec la souris).
- Effectuer des déplacements de caméra (haut/bas, gauche/droite).

Incomplet/non supporté

- La liste des objets graphiques n'est pas disponible à l'usager (liste d'affichage).
- Ouvrir, fermer, sauver, sauver sous un espace de représentation.
- Ouvrir fermer, sauver, sauver sous un plan .
- Ouvrir fermer, sauver, sauver sous un objet graphique.

- Modifier un objet graphique (échelle couleur, imprimable, affichable).
- Retirer un objet graphique, déplacer un objet graphique.
- Copier, coller, couper un objet graphique.

Bien qu'une partie du travail est complétée, le fait que la version livrée n'offre pas de liste d'affichage empêche de démontrer plusieurs fonctionnalités.

4.1.4.7 Outil d'analyse : visualisation

Supporté

- Charger des champs scalaires et vectoriels dans MayaVi et appliquer des filtres sur ceux-ci pour afficher des isolignes, des isosurfaces, des points, des vecteurs, etc.
- Visualiser des maillages.

Le virage vers MayaVi en tant qu'outil pour effectuer les tracés fait que les fonctionnalités décrites dans les spécifications ne sont plus tout à fait exactes.

L'utilisation de MayaVi n'est pas triviale au premier abord. La courbe d'apprentissage est abrupte si l'on veut maîtriser tous les paramètres VTK rendus disponibles par cet outil.

Pour Modeleur2, on rend disponible via le manuel de l'usager les explications pour réaliser certains affichages (ex : isolignes, isosurfaces, vecteurs, etc.), mais il est fortement recommandé de référer à la documentation de MayaVi.

4.1.4.8 Outil d'analyse : construction de séries

Supporté

- Créer, ouvrir, fermer, sauver et sauver sous des séries de réels ou de champs.
- Supporter les séries 1D et 2D.
- Modifier les coordonnées d'un axe d'une série manuellement
- Générer les coordonnées d'un axe d'une série à l'aide de pas.
- Éditer les valeurs portées par une série.

Incomplet/non supporté

- Modifier le type de données associé à un axe.

4.1.4.9 Outil d'analyse : scripts et macros

Supporté

- Exécuter des scripts Python.
- Utiliser un IDE externe pour l'édition de scripts (édition locale).
- Utiliser un débogueur externe pour le déboguage de scripts.
- Démarrer, terminer l'enregistrement de macros.
- Effectuer une pause lors de l'enregistrement.
- Exécuter une macro.

Incomplet/non supporté

- Importer un script d'une base de données globale.
- Exporter un script vers une base de données globale.

Il est important de mentionner qu'en raison d'un problème à l'enregistrement, très peu, voir aucun enregistrement de macros ne peut être rejoué directement avec succès. On peut par contre éditer les scripts des macros, les modifier manuellement puis les faire jouer.

4.1.4.10 Outil d'analyse : sonde

Supporté

- Ajouter un champ ou une série à la sonde.
- Modifier la projection SIG.
- Voir la valeur d'un champ ou d'une série à un point {x,y}.
- Modifier le mode (immédiat ou temps réel).

4.1.5 Librairies externes utilisées

Vu la gamme d'améliorations et d'ajouts à faire à Modeleur dans le cadre du projet SEGRI, il a fallu chercher un moyen de diminuer la banque de code à la charge du groupe.

Le projet repose donc, dans la mesure du possible, sur des librairies externes gratuites et disponibles sur Internet sous des licences « open source ». Le logiciel Modeleur2 a donc été construit en faisant appel à plusieurs de ces librairies ou composantes externes. Une description de chacune d'elles est offerte dans les pages qui suivent.

Boa Constructor

Boa Constructor est un IDE multi plateforme et un outil de développement de GUI pour wxPython. Il est écrit en Python et utilise la librairie wxPython. Il fonctionne sur la plupart des systèmes d'exploitation (MS Windows 95/98/NT/2000/XP, Linux/BSD/UNIX) et est distribué sous la licence GPL.

Le site http://boa-constructor.sourceforge.net/ offre de la documentation sur Boa Constructor.

Boost

Boost fournit des librairies C++ gratuites et portables qui ajoutent des fonctionnalités supplémentaires au C++. L'emphase est mise sur des librairies qui fonctionnent bien avec celles du standard C++. Les librairies Boost ont été développées pour être utilisées dans toute une gamme d'applications commerciales ou non.

Entre autres, la librairie Boost-Python offre une interopérabilité entre les langages de programmation C++ et Python. Elle fut abondamment utilisée tout au long du développement de Modeleur2 car beaucoup de structures C++ furent publiées en Python.

Boost vise à s'imposer comme étant le leader du développement C++ et ainsi fournir des implémentations de référence qui pourront éventuellement être incluses dans le standard C++. Plusieurs des librairies développées sont d'ailleurs en cours d'examen par le comité responsable du standard.

Bien qu'originalement commencé par les membres du comité du standard C++, la participation au développement des librairies s'étend désormais à des milliers de programmeurs C++.

Le site web http://www.boost.org/ offre de la documentation sur toutes les librairies proposées par Boost.

GEOS

La librairie GEOS (Geometry Engine - Open Source) est une traduction en C++ de la suite JTS (Java Topology Suite).

La librairie GEOS supporte la spécification OpenGIS ("Simple Features for SQL"). Elle est utilisée par PostGIS, l'extension SIG de PostgreSQL pour effectuer toutes les opérations géométriques. Toutes les fonctionnalités de requêtes spatiales repose donc en bout de ligne sur cette librairie.

La librairie GEOS est également utilisée dans les classes de géométrie du logiciel Modeleur2 pour effectuer diverses opérations géométriques.

Le site web http://geos.refractions.net offre de la documentation sur la librairie GEOS.

MayaVi

MayaVi est un module de visualisation de données scientifiques gratuit et facile à utiliser. Il est écrit en Python et utilise la librairie VTK pour l'affichage. MayaVi est écrit avec Tkinter. MayaVi fonctionne sur la plupart des plateformes où Python et VTK sont disponibles (Unix, Mac OSX ou Windows).

Le site web http://mayavi.sourceforge.net/ offre de la documentation sur MayaVi.

Ce logiciel est l'éditeur externe intégré à Modeleur2.

NTv2

NTV2 est un logiciel et contient des données d'interpolation des décalages du quadrillage permettant l'application d'une transformation nationale des coordonnées du NAD27 en coordonnées du NAD83.

Le site web http://www.geod.nrcan.gc.ca/software/ntv2_f.php offre de la documentation sur NTV2.

OTL

OTL est une librairie C++ multi plateforme qui implémente de façon générique le concept de flux OTL (otl_stream). Les flux OTL sont basés sur le même principe que les flux C++ qui fonctionnent habituellement avec les opérateurs >> et <<. La différence est qu'un flux OTL interagit avec l'API d'une base données pour effectuer la lecture/écriture de données, plutôt qu'un fichier, par exemple.

En plus de la fonctionnalité de flux, OTL offre entre autres la gestion des exceptions, les transactions, l'usage de paramètres liés (« bind parameters », de même que le support de la chaîne de caractères standard C++ (std ::string).

La version actuelle d'OTL offre un support natif vers Oracle et DB2, mais via l'interface ODBC elle permet l'utilisation des SGBD suivants : Oracle, MS SQL Server, Sybase, Informix, MySQL, DB2, Interbase / Firebird, PostgreSQL, SQLite, SAP/DB, TimesTen, MS ACCESS, etc.). La liste des engins de base de données supportés est en constante progression.

Le site web http://otl.sourceforge.net/ offre de la documentation sur la librairie OTL.

Pgstream

Pgstream est une librairie qui a été développée pour fournir un accès en C++ directement vers le serveur PostgreSQL. Pgstream ajoute une couche C++ au dessus de la librairie libpq (l'API pour interfacer PostgreSQL en langage C).

Comme OTL, elle offre la fonctionnalité de flux, la gestion des exceptions, les transactions, l'usage de paramètres liés (« bind parameters »), de même que le support de la chaîne de caractères standard C++ (std ::string).

Le site web http://www.manitou-mail.org/pgstream/ offre de la documentation sur la librairie pgstream. Le projet est également hébergé sur pgfoundry à l'adresse suivante http://pgfoundry.org/projects/pgstream/.

PostgreSQL

PostgreSQL est un système de gestion de bases de données relationnelles objet basé sur POSTGRES, Version 4.2, développé à l'université de Californie au département des sciences informatiques de Berkeley. POSTGRES a lancé de nombreux concepts rendus ensuite disponibles dans plusieurs systèmes de bases de données commerciales.

PostgreSQL est le fruit de plus de 15 ans de développement. Il est bâti sur une architecture robuste qui lui vaut une forte réputation de fiabilité et d'intégrité des données. PostgreSQL fonctionne sur la plupart des systèmes incluant Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), et Windows.

PostgreSQL supporte une grande partie du standard SQL:2003 tout en offrant de nombreuses fonctionnalités modernes : requêtes complexes, clés étrangères, déclencheurs (triggers), vues, intégrité des transactions, contrôle des accès simultanés (MVCC ou « multiversion concurrency control »). De plus, PostgreSQL peut être étendu de plusieurs façons par l'utilisateur, par exemple en ajoutant de nouveaux types de données, fonctions, opérateurs, fonctions d'agrégat, méthodes d'indexation, langages de procédure.

Et grâce à sa licence libérale, PostgreSQL peut être utilisé, modifié et distribué par tout le monde gratuitement quelque soit le but visé, qu'il soit privé, commercial ou académique.

PostgreSQL est le SGBD utilisé lors du développement de Modeleur2.

Le site web www.postgresql.org offre de la documentation sur PostgreSQL.

PostGIS

PostGIS ajoute un support spatial au SGBD PostgreSQL. PostGIS a été développé par Refractions Research comme un projet de recherche. PostGIS est sous la licence GNU (General Public License).

Le site web http://www.postgis.org offre de la documentation sur PostGIS.

Proj4

Proj4 est une librairie de projection cartographique originalement écrite par Gerald Evenden qui faisait alors partie du USGS (U.S. Geological Survey).

Le site web http://proj.maptools.org/ offre de la documentation sur Proj4.

Python, wxPython

Python est un langage de programmation interprété, multi-paradigme, ce qui signifie qu'il autorise la programmation impérative structurée, orientée objet, et fonctionnelle. Il est doté d'un typage dynamique, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions.

Une syntaxe simple, des types de données de haut niveau et des bibliothèques extensives rendent ce langage très attractif pour le développement rapide d'applications, comme langage d'extension d'applications, pour le développement de scripts, ou encore comme interface permettant de connecter des composants existants. Sa syntaxe facilite son apprentissage et en fait un candidat idéal pour l'apprentissage de la programmation orientée objet.

Le langage Python est placé sous une licence libre. L'interpréteur Python est aujourd'hui disponible sur la majorité des plates-formes existantes (Amiga, *BSD, GNU-Linux, AIX, Solaris, Mac OS, Windows).
WxPython est une extension à Python pour développer des interfaces graphiques. wxPython est une collection de modules Python réalisée sur la base des wxWidgets de wxWindows, un « framework » multi-plateforme écrit en C++.

Python et wxPython sont au cœur même du logiciel Modeleur2. Toute l'interface graphique (GUI) est construite en wxPython. Un très grand nombre de classes de Modeleur2 ont été également traduites en Python à l'aide de la librairie Boost-Python. Le langage Python remplace le langage de la calculatrice.

Le site web www.python.org offre de la documentation sur Python. Le site web http://www.wxpython.org/ offre pour sa part de la documentation sur wxPython.

VTK

Le «Visualization ToolKit» (VTK) est un logiciel avec code source libre et gratuit qui sert à effectuer de l'infographie 3D, du traitement d'image et de la visualisation. Le logiciel est utilisé par des milliers de chercheurs et de développeurs dans le monde. VTK est constitué de classe C++ et de plusieurs couches d'interface incluant TCL/TK, Java et Python. Un support professionnel et des produits pour VTK sont fournis par Kitware, Inc.

VTK supporte une grande variété d'algorithmes de visualisation et des techniques avancées.

Le design et l'implémentation de la librairie fut fortement influencé des principes orienté-objet. VTK a été installé et testé sur pratiquement chaque système Unix, PCs (Windows 98/ME/NT/2000/XP), et Mac OSX Jaguar ou plus récent.

VTK est au cœur des fonctionnalités de visualisation de Modeleur2. Tout ce qui touche l'affichage dans les fenêtres graphiques est fait à l'aide de VTK.

Le site web http://www.vtk.org/ offre de la documentation sur VTK.

XML Parser

La librairie XMLParser offre des fonctionnalités simples pour lire et écrire un fichier XML. L'analyseur syntaxique XML charge le fichier en mémoire et construit une structure en arbre représentant le fichier XML.

Le site web http://iridia.ulb.ac.be/~fvandenb/tools/xmlParser.html offre de la documentation sur XML Parser.

Modeleur2 utilise des fichiers à quelques endroits, notamment dans les filtres d'import/export et dans les filtres des points de terrain.

Winpdb

Winpdb est un débogueur Python qui supporte les points d'arrêts, les « threads multiples », et le déboguage intégré d'applications.

Le site web http://www.digitalpeers.com/pythondebugger/ offre de la documentation sur Winpdb.

Cet outil est le débogueur qui est intégré à Modeleur2 et qui permet de déboguer des scripts Python.

4.1.6 Outils externes utilisés

Plusieurs outils ont été utilisés en cours de projet pour réaliser les tâches de développement logiciel. Une liste de ces outils et une courte description est présentée dans les pages suivantes (en ordre alphabétique).

Borland C++ Development Suite 5.0

Borland C++ Development Suite 5.0 pour Windows 95 et Windows NT est une suite d'outils intégrés permettant de maximiser le développement : Borland C++ 5.0,

CodeGuard 32/16 pour la détection de fuites mémoire à l'exécution, PVCS pour protéger le code source, Install Shield pour créer des programmes d'installation.

Cette suite d'outils a été utilisée exclusivement pour concevoir et maintenir le traducteur responsable de la traduction de projets Modeleur 1 vers Modeleur2. On a dû utiliser ce compilateur car les librairies contenant le code pour interfacer avec les structures Modeleur 1.0 ont été construites avec ce compilateur.

Le site www.borland.com offre de la documentation sur les produits Borland.

Boost.Jam

Boost.Jam (BJam) est un outil qui permet de construire des modules logiciels. Il se veut un remplacement au « make » classique. BJam est basé sur FTJam (http://packages.debian.org/unstable/devel/ ftjam) qui lui était basé sur Perforce Jam (http://www.perforce.com/jam/jam.html).

BJam est portable et fonctionne sous UNIX, VMS, Mac et NT. La plupart des fichiers Jam (JamFiles) sont également portables.

Jam gère la dépendance entres les modules automatiquement et Jam permet de construire des projets d'envergures répartis dans de multiples.

Le site http://www.boost.org/tools/build/jam_src/ offre de la documentation sur Boost.Jam.

Case Studio 2

CASE Studio 2 est un outil de modélisation de bases de données pour de nombreux SGBD, notamment (Access, Firebird, MS SQL, MySQL, PostgreSQL, Oracle, etc.).

C'est un outil de conception abordable, simple et convivial qui permet de concevoir, développer et gérer les structures de bases de données. Il convient aux développeurs individuels de logiciels, aussi bien qu'aux PME. Il a été conçu pour rendre agréable et efficace le procédé de conception et de développement de structures de bases de données simples ou complexes.

Il offre entres autres les fonctionnalités suivantes : la gestion des versions, la rétro conception, l'utilisation de sous modèles, la génération de scripts, les schémas ERD et DFD.

Il permet un développement professionnel et rapide, un accroissement de la productivité, moins d'erreurs de développement et une maintenance très efficace.

Le logiciel Case Studio 2 a été utilisé pour modéliser la base de données. Il permet de générer le script SQL qui permet de construire la base de données lorsque l'usager crée un nouveau projet Modeleur2.

Le site web www.casestudio.com offre de la documentation sur Case Studio 2.

CVS, Tortoise CVS

CVS, ou « Concurrent Versioning System », est un système de contrôle de version. Les système de contrôle de version sont utilisés pour suivre et coordonner les changements du code source au sein d'une équipe de développement.

TortoiseCVS est un client graphique (front-end) destiné à rendre l'utilisation de CVS plus facile et plus intuitive. Il permet aux développeurs de travailler sur les fichiers contrôlés par CVS directement dans Windows Explorer™.

Un des principales inconvénients de CVS est l'interface en ligne de commande qui est fournie. Beaucoup de développeurs deviennent plus habitué aux environnements de développement intégrés graphique (IDEs). TortoiseCVS a pour but de fournir cet environnement pointe et clique de façon intelligente et intuitive.

Le site web http://www.cvs.org offre de la documentation sur CVS alors que le site web http://www.tortoise.org/ offre de la documentation sur Tortoise CVS.

L'outil Tortoise a été installé sur tous les postes développeurs. C'est l'outil qui fut utilisé pour interagir avec CVS.

Fast Bug Track

Fast BugTrack est un outil pour répertorier les bogues connus dans un logiciel et en faire le suivi. Fast BugTrack est disponible aux utilisateurs via un navigateur web. Il est facile à installer et facile à utiliser.

Le site web www.fastbugtrack.com offre de la documentation sur Fast BugTrack.

Cet outil a été utilisé lors du développement de Modeleur2 pour effectuer la gestion des bogues et sera également utilisé par les usagers externes via une page web lié avec le serveur de bogues.

GCC

GCC est le compilateur utilisé dans les systèmes GNU. Le développement de GCC se fait dans un environnement de développement ouvert et sur plusieurs plateformes. L'objectif est de livrer un compilateur de classe mondiale qui attire beaucoup de développeurs et de s'assurer que GCC fonctionne et soit testé sur de multiples architectures.

GCC (via cygwin) a été un des compilateurs utilisés lors de la compilation multi compilateurs. Ce compilateur est réputé pour suivre le standard C++ à la lettre et permet de trouver des erreurs que d'autres compilateurs ne traitent pas.

Le site web http://gcc.gnu.org/ offre de la documentation sur le compilateur gcc.

Intel C++

Intel C++ est un compilateur développé par la compagnie Intel. Il s'intègre notamment dans l'environnement de développement Visual Studio. Le compilateur Intel a été développé pour tirer profit de l'architecture Intel pour optimiser les performances.

Le site www.intel.com offre de la documentation sur Intel C++.

Intel C++ a été un des compilateurs utilisés lors de la compilation multi compilateurs. Ce compilateur s'intègre aisément dans l'environnement de développement Visual Studio.

pgAdmin

pgAdmin III est une plate-forme puissante d'admistration et de développement pour la base de données PostgreSQL, libre et gratuite pour tous usages. L'application fonctionne sous GNU/Linux, FreeBSD et Windows 2000/XP.

pgAdmin III a été conçu pour répondre aux besoins de tous les utilisateurs, depuis l'écriture de requêtes simples jusqu'au développement de bases de données complexes. L'interface graphique donne accès aux fonctionnalités de PostgreSQL les plus récentes, faisant de l'administration un exercice aisé. L'application comprend également un constructeur de requêtes, un éditeur SQL, un éditeur de code server-side et bien plus encore.

pgAdmin III est le fruit d'un projet collaboratif international réunissant des spécialistes des bases de données. L'application est disponible dans plus de 30 langues sous licence libre Artistic licence.

Cet outil livré avec PostgreSQL est utilisé pour visualiser le contenu des tables de la base de données. Il est très utile lorsque vient le temps de déboguer des sections de code qui implique la base de données.

PLOne / Zope

Plone est un outil de gestion du contenu (CMS) centré sur Zope. L'équipe de Plone comprend des experts en ergonomie web dont la contribution a permis l'avènement d'un outil attractif et qui facilite les tâches des gestionnaires de contenu : l'ajout, la mise à jour et la maintenance du contenu. Plone est standard et respecte rigoureusement les normes d'ergonomie et d'accessibilité et se base sur les meilleurs standards du web tels que XHTML et CSS.

Plone est disponible sous licence GPL (GNU General Public License), la même que celle utilisée par Linux. La communauté de développeurs Plone compte près d'une centaine de développeurs à travers le monde, et un nombre important de sociétés spécialisées dans le développement et le support autour de Plone. Il existe de nombreux composants qui permettent d'étendre Plone avec de nouvelles fonctionnalités et types de contenu. Plone peut interopérer avec la plupart des systèmes de bases de données relationnelles, open source ou propriétaires, et tourne sur la majorité des systèmes d'exploitation, y compris Linux, Windows

Le site web http://www.plone.org/ offre de la documentation sur Plone et le site web http://www.zope.org/ offre de la documentation sur Zope.

Plone / Zope a été utilisé pour construire le site web de Modeleur2 et son manuel de l'usager.

Visual Paradigm

Visual Paradigm-UML est un outil puissant de design UML et multi plateforme. L'outil est destiné à un grand éventail d'utilisateurs, incluant les développeurs, les analystes de système, les architectes de système qui sont tous intéressés à construire des systèmes de grande envergure en se basant sur l'approche orientée objet. Il permet de construire des applications de qualité, rapidement et à faible coût (la version « community » est disponible sans frais). Il facilite l'interopérabilité avec d'autres outils de design et plusieurs IDE.

Le site http://www.visual-paradigm.com/product/vpuml offre de la documentation sur Visual Paradigm UML.

Cet outil a été utilisé pour effectuer le design de plusieurs modules du logiciel. Il a principalement servi à réaliser des diagrammes de classes et de diagrammes de séquence.

Visual Studio .NET

Le système de développement Visual Studio .NET constitue l'environnement de développement intégré de Microsoft permettant de créer des applications sécuritaires et puissantes et des services Web en langage XML. Visual Studio .NET vise à accroître la productivité des développeurs et à les aider à bâtir et à intégrer rapidement des applications et des services Web, ce qui entraîne un meilleur temps d'accès au marché. Visual Studio .NET propose le compilateur Microsoft Visual C+.

Le site http://msdn.microsoft.com/vstudio/ offre de la documentation sur Visual Studio.

L'environnement de développement Visual Studio a été utilisé tout au long du projet pour développer Modeleur2.

Web++

Web++ est un outil de documentation qui permet d'analyser du code C++ et de produire des pages WWW représentant le code développé par Yves Secretan (INRS).

A l'aide d'hyper-liens, il est possible de se déplacer dans les pages générées et d'accéder à la page d'une classe en particulier. Pour chaque classe, Web++ produit : des liens vers les fichiers contenant le code, des liens vers le ou les parents, des liens vers le ou les héritiers, la liste des attributs, la liste des méthodes avec accès direct sur le code de la méthode et une vue colorée du code avec les liens vers les types connus.

Le site web http://www.gre-ehn.inrs-ete.uquebec.ca/download/webpp offre de la documentation sur cet outil.

Cet outil a été utilisé pour publier à l'interne une documentation conviviale pour naviguer à travers le code des différents modules du logiciel.

WinMerge

WinMerge permet de fusionner deux fichiers en comparant leurs différences. Il suffit d'ouvrir deux fichiers et le programme met en évidence les différences par des couleurs appropriées. Une fonction permet de mettre à jour un des fichiers à partir de l'autre. L'affichage est personnalisable et certaines chaînes peuvent être filtrées pour ne pas être concernées. L'outil est gratuit et Open Source.

Le site www.winmerge.org offre de la documentation sur WinMerge.

Cet outil fut utilisé pour effectuer des comparaisons entre des fichiers. Il fait ressortir les différences et permet de modifier le fichier.

4.2 H2D2

H2D2 est le successeur d'Hydrosim et de Dispersim (Hydrosim2-Dispersim2). Il s'appuie sur la dernière version d'Hydrosim et de Dispersim ainsi que sur divers développements annexes réalisés au cours de mandats antérieurs.

Hydrosim (Heniche *et al.*, 2000) est un modèle aux éléments finis qui résout les équations de St-Venant bidimensionnelles intégrées sur la verticale. Dispersim (Heniche *et al.*, 2001), pour sa part, est un modèle aux éléments finis de transport-diffusion qui s'appuie sur les résultats hydrodynamiques fournis par Hydrosim, et qui permet de traiter simultanément plusieurs cinétiques ainsi que leur interaction.

Ces deux logiciels ont fait leurs preuves dans des conditions très variées, que ce soit pour des applications faites par l'INRS ou par des partenaires et utilisateurs externes. Leur robustesse, et surtout pour Hydrosim, l'intégration étroite à Modeleur, ont mis les études

ayant une composante hydrodynamique à portée de main de non-spécialistes en hydraulique fluviale.

Les besoins exprimés par les utilisateurs touchent principalement deux aspects :

- la résolution de gros systèmes, en particulier la résolution répartie;
- l'ajout aisé de nouvelles fonctionnalités, notamment :
  la possibilité d'ajouter des algorithmes;
  la possibilité d'ajouter des méthodes de résolution;
  la possibilité d'ajouter des éléments;
  la possibilité de modifier des parties d'un élément, comme d'ajouter un traitement de conditions limites.

Si Hydrosim et Dispersim sont des logiciels puissants en terme de capacité de traitement, ils sont relativement limités dans leur extensibilité. Il est difficile, voire impossible de rajouter dans la structure actuelle, une méthode résolution, un post-traitement, des commandes dans l'interface etc.

4.2.1 Spécifications fonctionnelles

Les besoins exprimés par les utilisateurs touchent principalement les aspects qui sont développés ci-dessous.

4.2.1.1 La résolution de gros systèmes

Les plateformes 32 bit ont une capacité d'adressage limitée à 4GB. Le système d'exploitation, dans le meilleur des cas ne se réserve que 1GB, laissant donc au maximum 3GB pour les applications. Pour de gros problèmes, rigides ou difficiles à résoudre et qui demandent soit une résolution directe, soit une matrice de pré-conditionnement bien remplie, soit une base de Krylov plus large, ce n'est pas suffisant.

Il faut envisager de porter l'application sur des plateformes 64 bits. Ceux qui sont disponibles sont Linux et Windows. La transition aux plateformes 64 bits est un changement en profondeur qui force à valider toutes les déclarations de variables et le passage des paramètres ainsi que la compilation sur différents compilateurs et plateformes.

Les machines multiprocesseurs sont maintenant plus facilement disponibles et il serait intéressant de disposer de méthodes de résolution qui exploitent au mieux ces capacités. Les boucles de calculs intensifs doivent être identifiées, analysées (pour en faire ressortir les éléments à traiter en parallèle), puis codées en utilisant des librairies externes optimisées (BLAS2, BLAS3) et des directives de parallélisation du protocole OpenMP (OpenMP, 2005).

La procédure de parallélisation n'est possible que dans du code propriétaire, dans notre cas principalement sur la factorisation de matrices et les produits matrice-vecteur.

Il est également possible d'exploiter des grappes d'ordinateurs, donc plusieurs ordinateurs réunis par réseau pour résoudre un problème. Il y a ici plusieurs aspects :

- le partitionnement: la séparation du maillage en blocs répartis sur les machines;
- l'assemblage de la matrice globale à partir des contributions de chaque machine;
- la résolution du système assemblé;
- la redistribution du résultat sur chaque machine.

En ce qui a trait au partitionnement, les frontières qui sont des régions partagées entre plusieurs machines et qui doivent donc être synchronisées régulièrement doivent être minimales (afin de minimiser la communication entre les machines). On exploite également le fait que les partitions sont plus petites pour faire une renumérotation locale et ainsi optimiser la largeur de bande sur chacune des machines.

Le partitionnement utilise les librairies externes METIS et ParMETIS comme plug-in (module externes). La résolution utilise la librairie externe PETSc également comme plug-in. La communication entre les machines est faite à l'aide du protocole MPI (Message Passing Interface) et de la librairie externe MPICH2.

Tout le code doit être modifié pour prendre en compte les tables de partitionnement, pour distribuer les données à la lecture et à l'écriture, pour supporter une numérotation locale et une numérotation globale. Enfin, les librairies externes doivent être interfacées pour qu'il soit possible de profiter de leurs fonctionnalités.

4.2.1.2 L'ajout aisé de nouvelles fonctionnalités

A divers niveaux, on veut être capable :

- d'ajouter des algorithmes;
- d'ajouter des méthodes de résolution;
- d'ajouter des éléments;
- de modifier des parties d'un élément, comme d'ajouter un traitement de conditions limites.

4.2.1.3 Les éléments 1D1/2 et 2D1/2

Nous avons proposé de développer deux nouveaux éléments pour enrichir Hydrosim. Ils sont désignés 1D½ et 2D½. et sont obtenus respectivement en simplifiant le 2D traditionnel et en l'enrichissant. Le modèle alternatif 1D½ doit permettre de conserver la richesse de description de terrain tout en ayant une modélisation hydrodynamique allégée pour le cas où seuls les niveaux d'eau sont importants, par exemple pour traiter des problématiques d'inondation. Le modèle 2D½ doit permettre de conserver une description

de terrain fine tout en améliorant la description des vitesses sur la verticale et ceci sans payer le prix du 3D.

4.2.2 Architecture

4.2.2.1 La structure d'Hydrosim et de Dispersim

Si Hydrosim et Dispersim sont des logiciels puissants en terme de capacité de traitement, ils sont relativement limités dans leur extensibilité. Il est difficile, voire impossible de rajouter dans la structure actuelle une méthode résolution, un post-traitement, des commandes dans l'interface, etc.

En termes de structure informatique, Hydrosim et Dispersim sont modulaires au sens que cette notion avait dans les années 1980 et 1990. Ils partagent une Coquille logicielle commune qui comprend entres autres le traitement des commandes, la lecture des données et la résolution par une méthode de GMRes non-linéaire. La façon de faire actuelle repose plus sur une approche orienté-objet et des composantes (plug-in) avec une interface de programmation (API) bien définie.
Lors de la mise à niveau de Dispersim pour l'intégrer à la Coquille commune (Secretan et al., 2000), nous avons à l'époque utilisé des concepts de programmation orienté-objet, même si le Fortran77 est un langage purement procédural. Les diverses cinétiques des contaminants sont introduites comme une spécialisation par héritage d'une base commune qui, dans ce cas, s'occupe de toute la partie transport-diffusion (voir Figure 13).

Le bénéfice de cette approche ne se trouve pas tant en capacité de résoudre de plus gros problèmes ou d'une plus grande vitesse d'exécution, mais plutôt en réutilisation de code validé. L'ajout d'un nouvel élément (une nouvelle cinétique) est alors beaucoup plus simple, ce qui facilite l'extensibilité du logiciel.

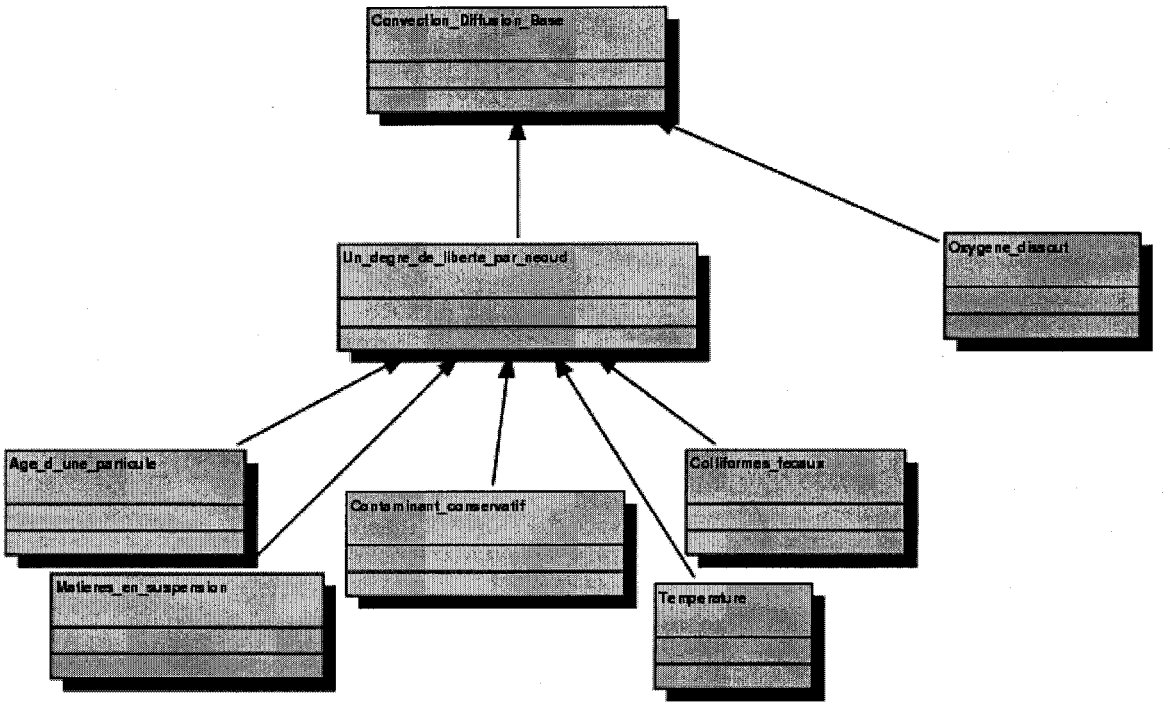


Figure 13 - Diagramme UML des classes de transport-diffusion.

4.2.2.2 La structure d'H2D2

H2D2 applique également aux éléments de St-Venant (Figure 14) le concept d'héritage présenté précédemment.

On récupère l'élément St-Venant d'Hydrosim tout en le restructurant quelque peu puisque cet élément était étroitement lié à la méthode de GMRes non-linéaire implantée dans la Coquille.

Puisque les fonctions en place y sont très spécialisées, il est nécessaire de les adapter pour les rendre plus génériques et ainsi pouvoir les utiliser avec d'autres méthodes de résolution.

Pour donner un exemple, Hydrosim n'assemble jamais la matrice de rigidité [K] car la méthode GMRes non-linéaire n'utilise que le produit [K]{u} qui est assemblé directement. Le stockage de la matrice et des produits matrice-vecteur est ainsi évité. Toutefois, les autres codes de résolution ont généralement besoin de manière explicite de cette matrice [K] (ex : PETSc, SuperLU, etc..). La classe de base St-Venant doit donc rendre disponible les fonctionnalités pour obtenir cette matrice.

A titre d'exemple, avec la nouvelle façon de faire basée sur l'héritage, une modification du traitement du couvrant-découvrant, n'implique pas de copier ou de réécrire l'élément mais uniquement de changer la partie concernée, et pour le reste de faire appel à son élément parent.

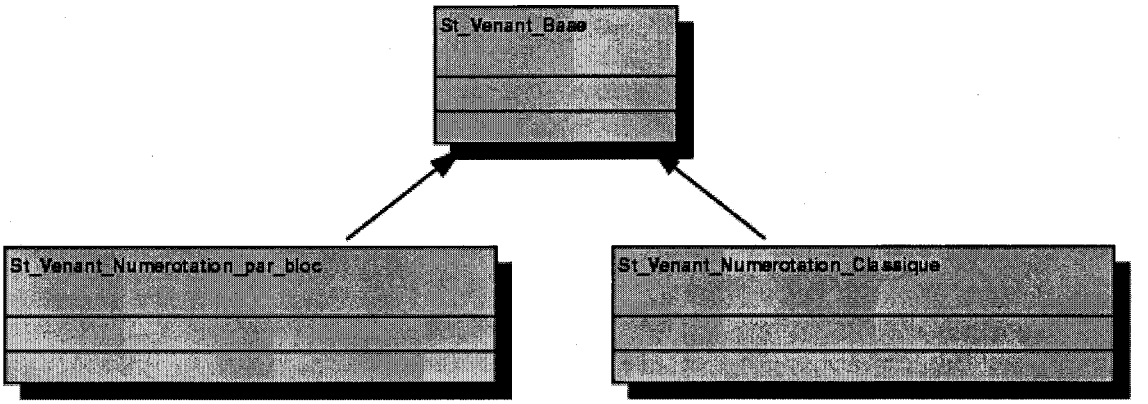


Figure 14 - Diagramme UML des classes de St-Venant 2D

Dans H2D2, on applique également cette approche orienté-objet aux autres composantes du logiciel et on définit les structures de données suivantes :

- les fichiers de données;
- les données de simulation (globales, nodales, élémentaires);
- les matrices;
- les méthodes de résolution matricielles;
- les algorithmes temporels, linéaires et non-linéaires;
- les post-traitements;
- les éléments géométriques.

La hiérarchie des algorithmes implantée actuellement dans H2D2 est schématisée à la Figure 15.

En pratique, à chacune de ces composantes correspond une commande qui permet de la créer et de spécifier ses paramètres. Par exemple :

```
h_resmat = ldu_memory()
h_newton = picard (h_resmat, 5, 1.0e-15)
```

construit un algorithme de Picard de 5 itérations, avec un critère de convergence de 10^{-15} et une méthode de résolution matricielle par factorisation LDU en mémoire.

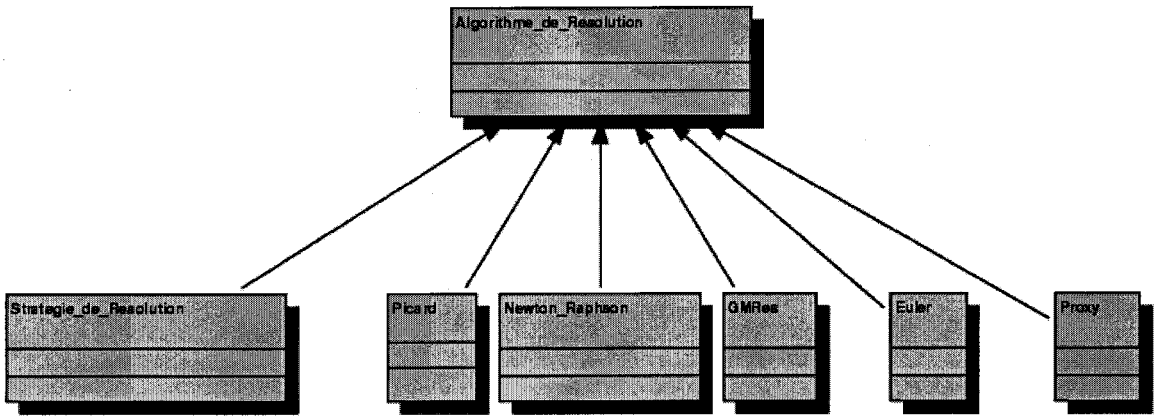


Figure 15 - Diagramme UML de la hiérarchie des algorithmes

4.2.2.3 Les proxys

La Figure 15 fait apparaître également la notion de classe proxy. Un proxy est une classe qui sert de relais à une composante définie à l'extérieur de H2D2. Sur un appel de fonction, le proxy fait appel à la même fonction de la composante externe. Cette composante externe doit bien entendu répondre à une interface de programmation, donc respecter un certain protocole et implanter les fonctions demandées par ce protocole.

Les composantes externes sont implantées sous forme de DLL et sont ajoutées à la configuration de H2D2 via un fichier de configuration. On peut ainsi ajouter :

- des commandes;
- des algorithmes, des solveurs, des pré-conditionneurs, des matrices;
- des éléments géométriques, des éléments de résolution et des conditions limites;
- des post-traitements et des partitionneurs.

4.2.3 Développement

La nouvelle architecture mise en place dans H2D2 et la souplesse dans l'ajout de composantes ont permis l'implantation des fonctionnalités suivantes:

- **Ajout de nouvelles méthodes de résolution**

  Les méthodes internes suivantes sont désormais supportées: GMRes, factorisation LDU en mémoire, factorisation LDU sur disque.

  Les librairies externes suivantes sont supportées via un plug-in : PETSc, SuperLU.

- **Optimisation des algorithmes**

  La factorisation LDU est parallélisée en utilisant la librairie OpenMP.

- **Calcul réparti**

  Pour être en mesure d'effectuer du calcul réparti, il a fallu qu'il soit possible d'introduire les tables de partitionnement et les numérotation locales et globales.

  Il a également été requis de brancher ParMETIS comme plug-in pour effectuer le partitionnement

- **Séparation des propriétés nodales**

  Les propriétés nodales peuvent provenir de fichiers différents.

  On peut avoir un fichier pour le terrain (topographie, coefficients de Manning, etc.), un pour la glace, un pour l'hydraulique (niveau d'eau, vitesses, etc.).

  Chaque fichier a son propre pas de temps, ce qui permet de mélanger une topographie invariante dans le temps avec une description des macrophytes évolutive.

Toutes les données sont maintenant prises dans le temps, les conditions stationnaires étant un cas particulier.

Nous avons également séparé les conditions aux limites en deux informations distinctes : les limites qui représentent la partie géométrique et les conditions portées par ces limites.

Nous avons donc, avec H2D2, un logiciel souple et extensible qui nous permet d'envisager traiter des gros problèmes et des problèmes non-stationnaires.

4.2.3.1 Banc de test

Les tests sont une étape critique de la validation d'un logiciel. Ils permettent de valider composante par composante la conformité des résultats produits par le code.

Le banc de test est actuellement constitué d'une centaine de tests individuels. Chacun de ses tests comprend des fichiers de configuration, des fichiers de commande et un fichier des résultats attendus avec les tolérances acceptées. On peut faire des tests sur une valeur, sur un norme d'erreur ou encore sur un taux de convergence. Ces tests sont exécutés automatiquement puis les résultats compilés sur une page web.

Le système de validation automatisé construit dans le cadre de Modeleur2 fut également utilisé pour construire et valider H2D2 de façon automatisée.

Cette infrastructure, qu'il faut évidemment maintenir, garantie que le code reste conforme aux attentes tout au long du développement.

4.2.4 État des logiciels

H2D2 est fonctionnel et a été intégré à Modeleur1. H2D2 a été utilisé pour des calculs instationnaires de transport-diffusion où la séparation des propriétés nodales sur des fichiers indépendants et la résolution parallélisée par LDU sur disque ont permis de résoudre de gros problèmes.

L'élément de St-Venant est programmé mais n'est pas encore opérationnel. Une comparaison serrée entre Hydrosim et H2D2 reste à faire, tant au niveau des taux de convergences que des résultats.

Au niveau du calcul réparti, si la programmation est achevée, il reste ici également à valider et tester cette partie pour déterminer si les gains attendus en terme de vitesse de calcul se réalisent.

Mme Giasson, étudiante à la maîtrise, est actuellement à implanter un élément 2D ½ .

Si le développement de H2D2 n'est pas terminé, c'est en parti en raison du fait que le recrutement d'étudiants est difficile. Les aptitudes exigées pour s'initier aux méthodes numériques sont élevées et le nombre de candidats est très limité. La recherche proposée qui contient un mélange d'éléments finis, d'hydraulique fluviale, de modèle numérique de terrain et parfois de biologie est difficile par sa multidisciplinarité intrinsèque, et effraie souvent les demandeurs potentiels.

Bien que le logiciel ne soit pas entièrement complété, nous possédons désormais en H2D2, une plateforme avec laquelle il est beaucoup plus facile de travailler et à laquelle il sera aisé d'ajouter des fonctionnalités.

4.2.5 Librairies externes utilisées

H2D2 fait appel à des librairies externes gratuites et disponibles sur Internet sous des licences « open source ».

BLAS2, BLAS3

BLAS est un librairie de routines appelables en Fortran hautement optimisées, et qui implémente le protocole BLAS (Basic Linear Algebra Subprograms). BLAS3 implante les opérations de matrice-matrice alors que BLAS2 implante les opérations matrice-vecteur et vecteur-vecteur.

Le site web http://www.csit.fsu.edu/~burkardt/f_src/blas3/blas3.html offre de la documentation sur les librairies BLAS2 et BLAS3.

Metis/ParMetis

ParMETIS est un librairie parallèle basée sur MPI qui implémente une variété d'algorithmes pour effectuer le partitionnement de graphes non structurés et de maillages, et pour traiter des matrices creuses. ParMETIS étend les fonctionnalités fournies par METIS et inclut des routines qui sont spécialement adaptées pour le calcul parallèle AMR (« Adaptive Mesh Refinement ») et pour les simulations numériques à grande échelle.

Le site http://www-users.cs.umn.edu/~karypis/metis/parmetis/ offre de la documentation sur ParMETIS. Le site http://www-users.cs.umn.edu/~karypis/metis/metis/index.html offre de la documentation sur METIS.

MPICH2

MPICH2 est une librairie portable et gratuite qui implémente le MPI (la librairie standard pour le passage de message). La documentation pour installer et utiliser MPICH2 est disponible aussi bien qu'un manuel pour les routines et les commandes.

Le site web http://www-unix.mcs.anl.gov/mpi/mpich2/ offre de la documentation sur MPICH2.

OpenMP

OpenMP est une interface de programmation (API) qui supporte la programmation parallèle multi-plateforme en C/C++ et Fortran sur toutes les architectures incluant Unix et Windows. Développé par un groupe incluant les plus importants vendeurs de matériels et de logiciels, OpenMP est un modèle portable qui offrent aux programmeurs une interface simple pour développer des applications parallèles.

Le site web http://www.openmp.org/drupal/ offre de la documentation sur OpenMP.

SuperLU

SuperLU est une librairie permettant la résolution de grand systèmes d'équations linéaires non symétriques sur des machines haute performance. La librairie est écrite en C et peut être utilisée à partir du C ou du Fortran.

Le site web http://www.cs.berkeley.edu/~demmel/SuperLU offre de la documentation sur SuperLU.

PETSc

PETSc est un ensemble de structures de données et de routines pour la solution parallèle et répartie d'applications scientifiques modélisées par des équations différentielles partielles. La librairie emploie le standard MPI pour toute la communication et le passage de message.

Le site http://www-unix.mcs.anl.gov/petsc/petsc-2 offre de la documentation sur la librairie PETSc.

4.2.6 Outils externes utilisés

Les outils CVS, Tortoise, Winmerge et Visual Studio qui ont été présentés dans la section traitant de Modeleur2 sont eux aussi utilisés dans le développement d'H2D2. Voici d'autres outils qui ont servi au développement d'H2D2.

Intel Fortran

Intel Fortran est un compilateur développé par la compagnie Intel. Il s'intègre notamment dans l'environnement de développement Visual Studio. Le compilateur tire profit des capacités des processeurs multicoeur et des systèmes multiprocesseurs.

Le site www.intel.com offre de la documentation sur Intel Fortran

4.3 Environnement de développement

Pour mener à bien un projet comme Modeleur2, il était important de mettre en place un environnement de développement particulier en ce qui a trait aux aspects organisation humaine et organisation physique. Ces deux aspects sont discutés dans les pages qui suivent.

4.3.1 Organisation humaine

Étant donné les ressources financières limitées dans le cadre du projet SEGRI, le développement s'est fait avec une équipe réduite. Le cœur de l'équipe est constitué de deux développeurs seniors auxquels se greffent des stagiaires (le nombre varie entre un et trois).

Modeleur 1.0 a été l'œuvre d'une équipe de six à huit personnes travaillant à plein temps sur le projet. Il est ici question d'un effort de 60 années-personne. Dans le projet Modeleur 2.0, l'ordre de grandeur de l'effort est plutôt de question de 8 à 10 années-homme (100-120 mois personne).

Pour compenser la diminution de la force de travail en comparaison avec Modeleur 1.0, il a donc fallu revisiter plusieurs des aspects du développement logiciel afin d'être en mesure de construire le logiciel, avec un niveau de qualité élevé et un jeu de fonctionnalités qui répond aux besoins du projet et aux attentes des partenaires et des utilisateurs.

Pour que le projet avance au rythme désiré, il est impératif que les stagiaires qui viennent travailler au sein du groupe laissent une contribution tangible et réutilisable. Le mot « réutilisable » est important ici car il faut bien voir qu'après le départ du stagiaire, il doit être aisé de reprendre le travail là où il l'a laissé et de continuer à avancer de manière quasi-transparente.

Un stagiaire demeure typiquement environ quatre mois au sein du groupe. Il a donc fallu mettre en place une structure adaptée à ce très haut taux de roulement. Dans cette optique, les mandats de stage sont découpés en tranches d'environ quatre mois. Pendant cette période, le stagiaire se voit confier un dossier complet (analyse, design, programmation, test) pour lequel il se voit remettre un échéancier. Il agit alors en tant que pilote. C'est à lui qu'il revient d'organiser son travail et de tenir des réunions d'analyse au besoin.

À chacune de ces réunions, il doit produire un rapport. Ces rapports de réunion sont importants car ils permettent de garder une trace des décisions qui ont été prises tout au long du dossier, et les raisons pour lesquelles elles ont été prises. À la fin de son stage, le stagiaire doit rédiger un rapport détaillant chacune des tâches effectuées dans le cadre de son dossier. Ce document sert par la suite de référence une fois que le stagiaire a quitté.

4.3.2 Organisation physique : système de validation et de test

4.3.2.1 Banc de test

Un banc de test a été développé pour valider la plupart des sections clés du logiciel. Le banc de test reproduit en quelque sorte ce que l'usager ferait à partir de l'interface graphique pour avoir accès aux fonctionnalités offertes par un module.

Les scripts de tests font en effet usage des mêmes événements que transmet l'interface graphique lorsque l'usager effectue une action (ex : un clic dans un menu). Il existe des tests pour valider les fonctionnalités du projet, des tests pour le MNT, pour les partitions de maillage et pour les séries. D'autres tests pourront évidemment être ajoutés à la banque de tests dans l'avenir.

Au moment d'écrire ces lignes, la batterie de tests s'effectue presque toute avec succès, mis à part quelques scripts qui font ressortir des bogues mineurs connus et qui n'ont aucun impact sur l'utilisation du logiciel en soi.

4.3.2.2 Système automatisé

Un système automatisé de validation du logiciel a été mis en place pour s'assurer de toujours avoir une base de code solide et fonctionnelle dans le gestionnaire de code source CVS (Concurrent Version System). C'est à partir de cette base fonctionnelle que les développeurs travaillent chaque jour.

Les développeurs peuvent ainsi ajouter leur contribution à cette base de code, et dès le lendemain, les ajouts effectués sont validés. Les erreurs, s'il y a lieu, peuvent donc être facilement identifiées.

Le système automatisé de validation est constitué d'une chaîne de processus qui chacun attend que le processus qui le précède dans la chaîne ait terminé sont travail.

La première étape consiste à mettre à jour à partir de CVS tous les modules nécessaires à la construction du logiciel. Par la suite, il y compilation des fichiers sources et reconstruction du logiciel pour une gamme élargie de compilateurs. Le système rend disponible les résultats de compilation sur le site intranet du groupe.

L'objectif de la compilation multi compilateurs est de valider que le code source produit répond avec exactitude au standard C++. En effet, il arrive que certains compilateurs ne détectent pas des erreurs alors que d'autres le font.

Au moment d'écrire ces lignes, tous les fichiers constituant le logiciel « compilent » et « link » sur Visual C++ 7.0, Intel 9.0 et GCC 3.3 en mode « debug » et « release ». Les fichiers compilent partiellement sur le compilateur VC 7.1 car étant donné la complexité du code, le compilateur n'arrive à compiler tout le code généré par Boost-Python.

Une fois la compilation terminée, les fichiers binaires produits sont placés sur la machine de distribution accompagné d'un fichier témoin. Un processus tourne sur cette machine, et dès qu'il détecte que le fichier témoin a changé, il génère les fichiers qui sont utilisés par l'outil d'installation. Ce processus produit à son tour un fichier témoin sur la machine responsable de l'exécution des tests.

Sur la machine de test, un processus tourne également et dès qu'il détecte que le fichier témoin a changé, il démarre automatiquement l'installateur pour une mise à jour. Les fichiers binaires, les exécutables et les scripts sur la machine de tests sont ainsi mis à jour.

Suite à cette opération, l'exécution du banc de test est démarré, ce qui produit en bout de ligne une page web indiquant les résultats des tests. À partir de la page de résultats, il est possible d'accéder aux fichiers de log produits par les tests.

4.3.2.3 Publication sur le site web

Les résultats de compilation quotidiens de même que tout ceux des compilations précédentes sont disponibles via une page web. Des liens sont fournis vers les fichiers d'erreur et d'avertissement pour chacun des fichiers compilé.

Une version de la documentation Web++ est générée quotidiennement à partir du code dans CVS et est rendue disponible via une page web. Le code hébergé sur le serveur CVS peut être consulté à partir d'une page web via l'outil ViewCVS.

Les résultats des tests sont eux aussi disponibles pour consultation via une page web. Des liens sont fournis vers les fichiers de log de chacun des scénarios de tests.

Au cours du développement, tous ces éléments étaient disponibles sur l'intranet. Ils le seront désormais sur une page web disponible à l'externe.

4.4 Distribution

La prochaine section discute d'aspects qui touchent tous les logiciels développés par le groupe. Il est notamment question de l'outil d'installation développé dans le cadre du projet, de la politique des licences et de la réingénierie du site web.

Une dernière section discute spécifiquement de la distribution de Modeleur2, seul logiciel actuellement disponible. H2D2 devrait être le prochain logiciel à bénéficier d'un site web.

4.4.1 Outil d'installation

Les logiciels sont distribués à l'aide d'un système similaire à Cygwin (émulation de la plateforme Unix sous Windows). Ce système permet la distribution et la mise à jour des différentes composantes du logiciel à partir du web.

L'outil de distribution Cygwin a été adapté et permet de télécharger les logiciels développés par le groupe. L'outil détecte les fichiers les plus récents et propose à l'usager de les mettre à jour sur sa machine. L'outil tient compte des dépendances qui peuvent exister entre les différentes composantes. L'utilisateur n'a jamais à tout réinstaller le logiciel manuellement. Lorsqu'il procède à une mise à jour, seules les parties qui ont changé sont réinstallées. En plus des fichiers propriétaires, l'outil peut installer toute une gamme de fichiers externes nécessaires à l'exécution des logiciels.

L'intérêt d'utiliser un tel système est de faire disparaître à toute fin pratique l'étape de distribution du logiciel.

4.4.2 Politique des licences et de la distribution des sources

Il été déterminé que la structure des licences telle qu'elle l'est actuellement devait être revue. Le passé a démontré que l'intérêt commercial se situe au niveau de services de consultation, plutôt que dans la vente des logiciels en tant que tel. Bien qu'aucune décision finale n'ait été prise à ce sujet, il apparaît fort probable qu'aucun montant d'argent ne soit exigé pour utiliser les logiciels.

Si, éventuellement, des plug-in sont développés avec des intérêts commerciaux, un système de licences pourrait être mis en place pour combler ce besoin.

De plus, puisque le logiciel est construit d'éléments qui font partie de la communauté « open source », il est légitime de penser que Modeleur2 devrait éventuellement être intégré à cette communauté.

Dans un premier temps, on rend disponible le code source du logiciel via l'outil Web++ pour consultation seulement. Le code source du logiciel pourra également être rendu disponible sur CVS à des développeurs externes sélectionnés.

4.4.3 Réingénierie du site web

L'architecture du site web a été revue et remodelée pour y intégrer un outil de gestion de contenu (CMS).

Il a été déterminé qu'un site web global présenterait les activités du groupe et qu'un site web serait également mis en opération pour chaque produit développé (i.e. chaque logiciel).

Un site produit contient une description du produit (i.e. logiciel), son manuel de l'usager, les instructions pour le télécharger et l'installer, une section contenant un forum de discussion, de même qu'une FAQ. Éventuellement, d'autres documents pourront être ajoutés au site d'un produit (ex : un tutoriel).

L'utilisation d'un outil de gestion de contenu a comme principal intérêt que la rédaction du site web peut-être faite par divers intervenants qui se voient attribués une gamme d'accès. Certains utilisateurs experts à l'intérieur de l'INRS (ou même de l'extérieur) se verront donc accorder des accès qui leur octroieront le droit de modifier la documentation. Il sera ainsi possible pour eux de maintenir la documentation à jour et d'ajouter des précisions s'il y a lieu de le faire. S'il est requis de garder un contrôle serré sur certaines sections, une validation pourra être exigé de la part des administrateurs.

L'outil qui fut sélectionné pour mettre en œuvre la nouvelle architecture après comparaison avec toute une gamme d'autres outils est Plone.

4.4.4 Site produit : Modeleur2

Modeleur2 est pour le moment le seul logiciel qui bénéficie d'un site web. Dans un premier temps, le site a été développé uniquement français. De par la structure du site web et en raison de la segmentation entre le contenu et le contenant, la traduction en langue anglaise devrait pouvoir se faire aisément.

Le guide de l'usager est disponible uniquement en format électronique à partir du site web, en français lui aussi.

Pour pouvoir utiliser toutes les fonctionnalités des logiciels (notamment Modeleur2), il est requis d'installer une base de données localement, ou encore d'installer un pilote ODBC pour interagir avec une base de données distante. Il est également requis d'installer le langage Python/wxPython. Enfin, il est suggéré d'installer l'outil Boa Constructor et l'outil Winpdb pour avoir accès aux fonctionnalités d'un IDE et d'un débogueur externe. Tous les autres éléments requis à l'utilisation de Modeleur2 seront installés par l'outil d'installation.

5 Ressources humaines

Les personnes suivantes ont été appelées à travailler sur le projet SEGRI durant la période allant de janvier 2003 à décembre 2005. La contribution de chacune de ces personnes est détaillée dans les prochaines pages.

Professeurs

Yves Secretan

M. Secretan, professeur à l'INRS-ETE, est responsable du projet SEGRI - volet informatique. Il a participé à la planification du projet et au recrutement de stagiaires. Il a travaillé à l'encadrement de ceux-ci et a participé à toutes les réunions de design et d'analyse tenues au cours des stages. Il agit également comme directeur de maîtrise de Maude Giasson.

En plus d'offrir un support aux stagiaires, M. Secretan a effectué diverses tâches de développement logiciel. Il a notamment développé H2D2, la version 2 des logiciels Hydrosim et Dispersim. Il a de plus travaillé sur la plupart des modules du logiciel afin de les améliorer ou de leur ajouter des fonctionnalités manquantes.

M. Secretan a travaillé à la rédaction des spécifications fonctionnelles, et à la révision de nombreux rapports ou documents. Il a également été impliqué dans la rédaction de plusieurs rapports ou articles qui furent publiés ou sont en voie de l'être (voir la bibliographie pour la liste exhaustive des documents rédigés).

Michel Leclerc

M. Leclerc, professeur à l'INRS-ETE, est responsable du projet SEGRI – volet application. Il fut directeur de maîtrise de Laurent Bonnifait.

M. Leclerc a été impliqué dans la rédaction de plusieurs rapports ou articles qui furent publiés ou sont en voie de l'être (voir la bibliographie pour la liste exhaustive des documents rédigés).

Il a notamment produit un important rapport traitant de l'application sur la Rivière Châteauguay.

Professionnels

Paul Boudreau

M. Boudreau, agent de recherche, a travaillé sur le volet application du projet. M. Boudreau possède une riche expérience avec Modeleur 1.0. Dans le cadre du projet, il a travaillé sur les applications de la Rivière Chaudière et de la Rivière Châteauguay. Il a contribué de près au rapport sur la Rivière Châteauguay rédigé par M. Leclerc.

M. Boudreau a également révisé les spécifications fonctionnelles et fourni des suggestions d'amélioration du logiciel.

Eric Larouche

M. Larouche, ingénieur en informatique, a comme tâche d'agir en tant que développeur senior au sein du groupe. Il a participé à la planification du projet et au recrutement de stagiaires. Il a travaillé à l'encadrement de ceux-ci et a participé à toutes les réunions de design et d'analyse tenues au cours des stages.

En plus d'offrir un support aux stagiaires, M. Larouche a effectué diverses tâches de développement logiciel. Il a travaillé sur la plupart des modules du logiciel afin de les améliorer ou de leur ajouter des fonctionnalités manquantes

M. Larouche a travaillé à la rédaction des spécifications fonctionnelles et des rapports officiels. Il a également travaillé à la rédaction / révision de plusieurs documents d'analyse, documents de génie logiciel ou de rapports.

Nicolas Roy

M. Roy, assistant de recherche, a effectué une grande part du travail de géomatique, de cartographie et de gestion des données. Il a également pris part à des campagnes de terrain. Il a contribué de près au rapport sur la Rivière Châteauguay rédigé par M. Leclerc.

Étudiants à la maîtrise (en ordre chronologique)

Maude Giasson

Mme Giasson a débuté sa maîtrise à l'automne 2004. Elle avait été originalement été recrutée pour travailler sur des méthodes pouvant aider à la validation des données provenant de levées topographiques au laser aéroporté.

Son mandat a toutefois été modifié suite au départ de M. Rahman (voir plus bas). Mme Giasson travaille désormais sur le modèle de simulation 1D ½. Elle est appelée à travailler avec le langage Fortran et à utiliser ses connaissances en mathématiques.

Khalid Rahman

M. Rahman a débuté une maîtrise à l'automne 2004. M. Rahman avait été recruté pour travailler sur le modèle de simulation 1D ½.

Il est extrêmement difficile de recruter des candidats qualifiés pour travailler dans le domaine des éléments finis et de la modélisation numérique principalement dû au niveau de difficulté élevé de ce champ d'étude.

M. Rahman semblait avoir le bagage et les capacités pour relever le défi. Il n'a malheureusement pas su s'adapter à la langue et n'arrivait pas à suivre le rythme requis par la maîtrise en raison de ses connaissances techniques insuffisantes.

M. Rahman a donc été remercié de ses services en début d'année 2005. Le mandat de développer le modèle de simulation 1D ½ a donc été transféré à Maude Giasson, elle aussi étudiante à la maîtrise. Le bagage de Mme Giasson allie les mathématiques et l'informatique. Elle se veut une candidate de choix pour poursuivre ce mandat.

Laurent Bonnifait

M. Bonnifait, étudiant à la maîtrise à l'INRS-ETE, a travaillé sur le projet SEGRI de juin 2003 à juillet 2004.

M. Bonnifait a effectué une revue de la littérature disponible au sujet de la ville de Ste-Marie de Beauce. Il a procédé à l'acquisition d'un échosondeur et à sa mise au point. Il a mené un campagne de terrain visant à recueillir des données sur la rivière Chaudière. Suite à cette campagne de terrain, il a intégré les données recueillies au Modèle Numérique de Terrain de la Rivière Chaudière. M. Bonnifait a également procédé à la rédaction de son mémoire de maîtrise intitulé « Développement de courbes submersion-dommages pour l'habitat résidentiel québécois ».

ML2005 : Y a t-il d'autres étudiants dont on voudrait faire mention dans le rapport?

Stagiaires (du plus récent au plus ancien)

Aurélien Mercier

M. Mercier, étudiant à l'École Pour l'Informatique et les Techniques Avancées (EPITA, Paris, France), a travaillé sur le projet SEGRI à titre de stagiaire en génie informatique. Son stage s'est effectué du 12 janvier au 9 juillet 2004.

M. Mercier s'est vu confié comme principal mandat de développer le module d'import/export. Il s'est également vu confié diverses tâches de développement visant à compléter certains modules ou en améliorer d'autres.

Sybil Christen

Mme Christen, étudiante à l'Université de Montréal, a travaillé sur le projet SEGRI à titre de stagiaire en informatique. Mme Christen est titulaire d'un baccalauréat en biologie (1994). Elle a également travaillé de 1996-2002 pour la firme Genilog en tant que programmeure-analyste.

Son stage à l'INRS-ETE s'est effectué du 16 mai au 26 août 2005. Mme Christen a également poursuivi son travail à titre de contractuelle du 12 septembre au 21 décembre 2005.

M. Christen s'est vu confié comme principal mandat de poursuivre le travail sur les séries débuté en 2003 par M. Kaczor. Elle a développé un outil permettant la construction et la gestion des séries. Elle a également développé l'outil de sonde.

Sébastien Quessy

M. Quessy, étudiant à l'Université du Québec à Trois-Rivières, a travaillé sur le projet SEGRI à titre de stagiaire en informatique. Son stage devait s'effectuer du 6 septembre au 16 décembre 2005. M. Quessy a mis fin à son stage de son propre chef le 14 octobre 2005.

M. Quessy s'est vu confié diverses tâches de développement visant à compléter certaines modules ou en améliorer d'autres. Il a notamment travaillé sur le module gestionnaire BD.

Olivier Bédard

M. Bédard, étudiant à l'Université Laval, a travaillé sur le projet SEGRI à titre de stagiaire en informatique. Son stage s'est effectué du 2 mai au 12 août 2005.

M. Bédard s'est vu confié diverses tâches de développement visant à compléter certains modules ou en améliorer d'autres. Il a notamment travaillé sur les dossiers suivants : module visualisation, module gestionnaire BD.

Thierry Malo

M. Malo, étudiant en génie des matériaux à l'École Polytechnique de Montréal, a travaillé sur le projet SEGRIS à titre de stagiaire en informatique. Son stage s'est effectué du 9 mai au 12 août 2005.

M. Malo s'est vu confié un mandat visant à développer un banc de test et des scénarios de test pour mettre à l'épreuve les divers modules qui composent Modeleur2.

Xavier Lavoie

M. Lavoie, étudiant au Cégep de Lévis-Lauzon, a travaillé sur le projet SEGRI à titre de stagiaire technicien en informatique. Son stage s'est effectué du 21 mars au 13 mai 2005.

M. Lavoie s'est vu confié comme principal mandat de standardiser les messages d'erreurs dans Modeleur2. Il s'est également vu confier divers autres tâches de programmation.

Jean-Philippe Lemieux

M. Lemieux, étudiant à l'Université de Sherbrooke, a travaillé sur le projet SEGRI à titre de stagiaire en informatique. Son stage s'est effectué du 10 janvier au 22 avril 2005.

M. Lemieux s'est vu confié comme principal mandat d'effectuer le développement du module de simulation.

Sébastien Labbé

M. Labbé, étudiant à l'Université de Sherbrooke, a travaillé sur le projet SEGRI à titre de stagiaire en mathématiques. Son stage s'est effectué du 7 septembre au 23 décembre 2004.

M. Labbé s'est vu confié comme principal mandat de continuer le développement des module MNT et Partition de maillage. Il a également élaboré et mis en place un algorithme pour effectuer les tâches de maillage qui n'est pas sensible aux problèmes de précision géométrique.

Vincent Martineau

M. Martineau, étudiant à l'Université de Sherbrooke, a travaillé sur le projet SEGRI à titre de stagiaire en informatique. Son stage s'est effectué du 7 septembre au 23 décembre 2004.

M. Martineau s'est vu confié diverses tâches de développement visant à compléter certains modules ou en améliorer d'autres. Il a notamment travaillé sur les dossiers suivants : module validation, module affichage, module interface graphique.

Stéphane Lévesque

M. Lévesque, étudiant à l'Université Laval, a travaillé sur le projet SEGRI à titre de stagiaire en informatique. Son stage s'est effectué du 10 mai au 13 août 2004.

M. Lévesque s'est vu confié un dossier visant à mettre en place une première phase du gestionnaire de données ainsi que la notion de projet. M. Lévesque a travaillé de près avec la base de données pour implanter les schémas, les requêtes de listes de données et les méta données. Il a également mis en place une série de boîtes de dialogue préliminaires.

Maude Giasson

Mme Giasson, étudiante à l'Université Laval, a travaillé sur le projet SEGRI à titre de stagiaire au Certificat en informatique. Mme Giasson est également titulaire d'un baccalauréat en mathématiques. Son stage s'est effectué du 5 janvier au 13 août 2004.

Mme Giasson s'est vu confiée comme principal mandat de mettre en place le module partition de maillage et d'y implanter la fonctionnalité permettant de mailler des couches. Dans le cadre de son mandat, elle a eu à travailler de près avec les modules d'affichage et de visualisation.. Elle a également eu à mettre en place le SRChampVector, un objet container de géométries et porteur de valeurs. Mme Giasson a débuté une maîtrise à l'automne 2004.

Daniel Nadeau

M. Nadeau, étudiant à l'Université Laval, a travaillé sur le projet SEGRI à titre de stagiaire en génie physique. Son stage s'est déroulé du 26 avril au 30 juillet 2004.

M. Nadeau s'est vu confié comme mandat de valider le modèle de températures avec le logiciel H2D2. Il a somme toute agit comme premier utilisateur officiel et a permis de tester les fonctionnalités de base du logiciel.

Benjamin Behaghel

M. Behaghel, étudiant à l'École Pour l'Informatique et les Techniques Avancées (EPITA, Paris, France), a travaillé sur le projet SEGRI à titre de stagiaire en génie informatique. Son stage s'est effectué du 12 janvier au 9 juillet 2004.

M Behaghel a travaillé sur le module de validation et le module de filtrage des données. Il a également développé un éditeur de structures géométriques qui est réutilisé à plusieurs endroits dans le logiciel.

Dave Guérin

M. Guérin, étudiant au Cégep de Lévis-Lauzon, a travaillé sur le projet SEGRI à titre de stagiaire technicien en informatique. Son stage s'est effectué du 29 mars au 14 mai 2004.

M. Guérin a travaillé au rodage du système de compilation automatisé. Il a éliminé toute une gamme d'erreurs et d'avertissements qui survenaient sur certains compilateurs.

Cédric Caron

M. Caron, étudiant à l'Université de Sherbrooke, a travaillé sur le projet SEGRI à titre de stagiaire en génie informatique. Son stage s'est effectué du 5 janvier au 16 avril 2004.

M. Caron s'est vu confié comme principal mandat de mettre en place le module MNT. Il a également travaillé à la mise en place des algorithmes du gestionnaire de données pour effectuer la sauvegarde et le chargement des champs et des séries.

Kevin Solinski

M. Solinski, étudiant à l'ENIC Telecom (France), a travaillé sur le projet SEGRI à titre de stagiaire en génie informatique. Son stage a débuté le 4 août 2003 et s'est terminé le 30 janvier 2004.

M. Solinski s'est vu confié un dossier traitant du module de script. M. Solinski a effectué une recherche afin de trouver un langage de script pour Modeleur 2.0. M. Solinski a travaillé à l'élaboration des spécifications fonctionnelles pour le module de script. Il a également travaillé à l'implémentation de celui-ci. Enfin, il a rédigé un rapport décrivant en détails les tâches réalisées.

Olivier Kaczor

M. Kaczor, étudiant à l'Université de Sherbrooke, a travaillé sur le projet SEGRI à titre de stagiaire en informatique. Son stage s'est effectué du 2 septembre au 19 décembre 2003.

M. Kaczor s'est vu confié un mandat traitant des champs et des séries. M. Kaczor a amélioré le traducteur pour y ajouter la traduction des champs. M. Kaczor a mené le design des champs, des séries et des itérateurs spatiaux, design qu'il a ensuite implémenté. M. Kaczor a également effectué une recherche pour trouver une librairie géométrique. Enfin, il a rédigé un rapport décrivant en détails les tâches réalisées.

Maxime Derenne

M. Derenne, étudiant à l'Université de Sherbrooke, a travaillé sur le projet SEGRI à titre de stagiaire en informatique. Son stage s'est effectué du 5 mai au 22 août 2003.

M. Derenne s'est vu confié un dossier traitant de la visualisation. Il a évalué des librairies graphiques et a effectué une recommandation. Il a mené le design de l'architecture du module de visualisation, qu'il a ensuite implémenté. Il a construit un prototype en Visual Basic qui intègre le module de visualisation et le module de gestion de données. Enfin, il a rédigé un rapport décrivant en détails les tâches réalisées.

Samuel Ouellet

M. Ouellet, étudiant à l'Université de Sherbrooke, a travaillé sur le projet SEGRI à titre de stagiaire en génie informatique. Son stage s'est effectué du 5 mai au 22 août 2003.

M. Ouellet s'est vu confié un dossier traitant de la base de données relationnelle. Il a évalué des outils de design et des engins de base de données. Il a ensuite effectué une recommandation. Il a mené le design des tables de la base de données relationnelle. Il a installé PostgreSQL / PostGIS et a rédigé un document explicatif. Il a démarré un projet de traducteur, dans lequel il a implanté la traduction des maillages. Enfin, il a rédigé un rapport décrivant en détails les tâches réalisées.

Sébastien Bédard

M. Bédard, étudiant à l'Université Laval, a travaillé sur le projet SEGRI à titre de stagiaire en génie mécanique. Son stage s'est effectué du 5 mai au 15 août 2003.

M. Bédard a travaillé à la conception d'un banc de test pour le logiciel H2D2. Pendant cette phase, il a également apporté différentes corrections au logiciel. M. Bédard a travaillé à la validation du modèle de température d'H2D2. M. Bédard a construit une ébauche de manuel utilisateur pour H2D2. Enfin, il a rédigé un rapport décrivant en détails les tâches réalisées.

Dominic Richard

M. Richard, étudiant au Cégep de Lévis-Lauzon, a travaillé sur le projet SEGRI à titre de stagiaire technicien en informatique. Son stage s'est effectué du 31 mars au 11 juillet 2003.

M. Richard a travaillé à la migration de certaines parties du code de Modeleur 1.0 vers des compilateurs plus récents. Avec ce code nettoyé, M. Richard a construit un certain nombre de modules de base qui sont réutilisés dans Modeleur2.

Renaud le Boulleur de Courlon

M. le Boulleur de Courlon, étudiant à l'Institut National Polytechnique de Toulouse (INP ENSEEIHT), a travaillé sur le projet SEGRI à titre de stagiaire en génie mécanique. Son stage s'est effectué du 14 février au 14 juin 2003.

M. le Boulleur de Courlon a travaillé à la cueillette des données sur la rivière Chaudière. Il a ensuite monté le Modèle Numérique de Terrain (MNT), construit un maillage et effectué des simulations.

Francis Larrivée

M. Larrivée, étudiant au Cégep de François-Xavier-Garneau, a travaillé sur le projet SEGRI à titre de stagiaire technicien en informatique. Son stage s'est effectué du 3 mars au 30 mai 2003.

M. Larrivée a travaillé à la recherche et à l'évaluation de solutions afin de mettre en place un système de compilation automatisé. M. Larrivée a également accompli diverses tâches d'installation / configuration de machines reliées à la mise et à jour du parc informatique.

6 Conclusion

Le projet SEGRI avait comme résultats attendus:

- une méthodologie d'analyse et de réduction des risques d'inondation appliquée et validée sur une rivière ;
- les logiciels qui soutiennent cette méthodologie.

L'aspect application scientifique et l'aspect développement logiciel sont discutés tour à tour dans les prochaines pages.

6.1 L'application scientifique

6.1.1 La méthodologie

Voici en résumé la méthodologie pour l'analyse et la réduction des risques d'inondation qui fut validée et développée au cours du projet.

Tenir compte de la portée réelle des coûts du risque

La portée du risque est étendue aux dommages d'incertitude sous la forme de leurs implications fiscales, et une méthode géomatique d'estimation des dommages directs est utilisée pour la prédiction du résultat des interventions.

Construire une base de données sur les aléas de crue et d'embâcles

Une base de données des aléas de crues et d'embâcles est constituée en mettant l'accent sur la série des maximums annuels horaires de débit si nécessaire, sur les facteurs conjoints (ex : niveaux maximums du fleuve), et sur l'ensemble des données événementielles d'embâcles disponibles chez les divers intervenants.

Établir la probabilité actuelle et évolutive des aléas

Une sélection des méthodes les plus appropriées est effectuée pour déterminer la probabilité théorique et/ou empirique, individuelle ou conjointe, des différentes classes d'aléas causant les inondations, dans les conditions actuelles, en présence d'aménagements remédiateurs ou en fonction des changements climatiques anticipés.

Simuler les conditions hydrauliques d'inondation actuelles et potentielles

La simulation numérique 1D et 2D des conditions d'écoulement actuelles et des divers scénarios remédiateurs est retenue. Le but de l'activité est de calculer la submersion due aux inondations pour chaque scénario étudié, et subséquemment les dommages directs.

Regrouper les données disponibles sur la vulnérabilité des constructions

L'activité vise la mise en place d'une base de données résidentielle avec comme variables les déterminants de la vulnérabilité aux dommages.

Établir la distribution des dommages et du risque

Une méthode géoréférencée d'estimation des dommages directs basée sur la submersion simulée et sur la vulnérabilité des bâtiments existants est appliquée. Pondérés par leur fréquence événementielle relative, les résultats constituent le risque de dommage direct (moyenne annuelle). Le dommage d'incertitude est estimé selon la perte de valeur à long terme et sa répercussion sur l'assiette fiscale municipale et scolaire. La question des refoulements n'est pas traitée bien qu'elle puisse constituer une part significative des dommages directs.

Rechercher des solutions et évaluer leur efficacité technique

Une caractérisation sommaire des différents types de solutions remédiatrices et leur analyse d'efficacité pour le contrôle du risque sont réalisées. L'accent est mis soit sur la réduction du niveau d'eau, ou sur la gestion de la vulnérabilité du parc résidentiel. L'analyse relative aux moyens administratifs vise surtout à identifier un rationnel permettant de désigner les propriétés qui pourraient en priorité être considérées pour une relocalisation. Une approche économique axée sur le rapport coûts/bénéfices est appliquée lorsque c'est possible.

Proposer une stratégie intégrée

Une (ou des) stratégie(s) globale(s) combinant les solutions identifiées en pré-faisabilité est recherchée en respectant quelques principes de base comme la rentabilité économique, l'intégration de toutes les composantes du risque et l'équité entre les intervenants. Le critère de réceptivité des solutions par les populations riveraines n'est pas évalué dans le cadre de cette étude.

6.1.2 Résultats obtenus (Rivière Châteauguay)

L'application réalisée à Châteauguay a permis de mettre à l'épreuve tous les éléments de la méthodologie proposée. Parmi les innovations, on compte :

- L'application conjointe du module de glace de HEC-RAS (US Army Corps of Engineers) et de Modeleur 1.0 afin de simuler les embâcles historiques et d'analyser l'influence du débit sur le niveau d'eau amont en situation d'embâcle.

- L'analyse de l'impact de l'implantation de structures de rétention de la glace en amont des secteurs vulnérables.

- L'analyse de solutions visant un accroissement de l'hydraulicité du cours d'eau basées sur le rétablissement d'anciens chenaux empiétés par le réseau de la voirie local.

- L'application de méthodes statistiques basées sur les probabilités conjointes afin de déterminer l'effet conjugué de plus d'un facteur de submersion.

- L'inclusion des embâcles dans le calcul de risque.

- La cartographie du risque unitaire d'endommagement (taux d'endommagement) en fonction des divers modes d'implantation résidentielle en zone submersible (ex; hauteur du rez-de-chaussée, présence et état de finition du sous-sol) et des paramètres statistiques et hydrauliques des aléas (crues à l'eau libre et embâcles confondus).

- L'analyse des opportunités de relocalisations et/ou immunisations.

- L'intégration du schéma de solution proposé dans une vision élargie de la vocation des cours d'eau en milieu urbain (accessibilité au plus grand nombre, par linéaire, etc.).

- La prise en compte des dommages d'incertitude et leur impact sur la fiscalité municipale, dans une perspective de développement rentable et durable.

Le rapport Leclerc et al., (2006) est en voie d'achèvement et il est appelé à faire école dans le domaine de la prévention des risques d'inondations à cause de la perspective élargie qu'il offre pour le résolution des problématiques les plus lourdes. Le lecteur est renvoyé à ce rapport pour de plus amples détails d'application.

6.2 Les logiciels

Les logiciels construits au cours du projet SEGRI sont Modeleur2 et H2D2. Modeleur2 est le remplaçant de Modeleur 1.0 alors qu'H2D2 est le remplaçant de Hydrosim et de Dispersim. Environ 90 % des efforts de développement ont été réalisés sur Modeleur2.

Les spécifications fonctionnelles de Modeleur2 (Secretan et al. 2004) présentent les fonctionnalités de Modeleur2 et discutent de son architecture.

6.2.1 Objectifs

Plusieurs objectifs expliquent l'effort de développement logiciel qui fut mis en branle pour construire le successeur de Modeleur 1.0. En voici une liste non exhaustive:

- offrir une ouverture vers l'extérieur permettant l'ajout de fonctionnalités et les contributions externes une fois une version officielle livrée;

- faire appel à un langage de script existant sur le marché de façon à offrir un maximum de fonctionnalités aux usagers;

- être en mesure de gérer et d'afficher les jeux de données de grande taille (nécessaire avec l'avènement des campagnes de terrain par laser aéroporté) ;

- proposer une interface utilisateur et un mode de fonctionnement intuitif pour les nouveaux usagers et efficace pour les usagers expérimentés;

- rendre les données disponibles aux utilisateurs via des moyens standards pour qu'il soit plus facile d'interagir avec les données du logiciel de l'extérieur;

- offrir des mécanismes pour procéder au filtrage automatisé des semis de points;

- mettre en place le concept des séries pour permettre entre autres de faire des calculs sur des champs interpolés à partir d'une série.

- proposer un mode de navigation interactif (déplacement/zoom), de même que des fonctionnalités d'affichage 3D, tout en tentant de réduire considérablement le code propriétaire au logiciel;

- implanter un mécanisme de maillage adaptatif où l'opération de maillage se fait à partir d'un champ d'erreur et d'un niveau de tolérance à l'erreur;

- être en mesure d'ajouter de nouveaux simulateurs (avec leur exécutable, leur interface graphique et tout ce qu'ils peuvent nécessiter);

- mettre en place un nouveau mécanisme de construction des partitions pour combler une lacune de son prédécesseur (lourdeur des contrôles de cohérence géométrique);

6.2.2 Décisions techniques clé

Construire un logiciel scientifique de l'envergure de Modeleur2 avec une équipe de développement constituée de deux développeurs senior et de stagiaires était un défi de taille. Plusieurs décisions techniques clé ont été prises pour y arriver, notamment:

- Miser sur la flexibilité et l'ouverture plutôt que tenter de tout faire;
- Miser sur des librairies externes;
- Utiliser le langage Python comme plaque tournante du logiciel;
- Effectuer le passage vers une base de données relationnelle;
- Utiliser VTK et MayaVi pour les représentations graphiques;
- Mettre en place un système automatisé de validation du logiciel.

Miser sur la flexibilité et l'ouverture plutôt que tenter de tout faire

Le logiciel propose le concept de plug-in. Toutes les fonctionnalités offertes par Modeleur2 le sont via le mécanisme des plug-in. Avec cette façon de faire, il sera aisé de compléter ou d'améliorer le logiciel dans l'avenir.

Un effort particulier a également été investi pour offrir des mécanismes pour ajouter des mailleurs, des simulateurs et des filtres.

Miser sur des librairies externes

Le logiciel a été construit à l'aide de librairies ou de composantes externes afin de profiter d'expertise externe et de code validé. Ceci permet également de diminuer de beaucoup la quantité de code à la charge du groupe et d'ainsi concentrer les efforts de développement sur ce qui est propriétaire à Modeleur2. Les librairies ou composantes externes suivantes sont utilisées dans les logiciels : Python/wxPython, PostgreSQL, PostGIS, VTK, MayaVi, GEOS, Boost, Boa Constructor, Winpdb, XML Parser.

Utiliser le langage Python comme plaque tournante du logiciel

Modeleur2 est construit autour du langage de script Python qui offre toute la flexibilité recherchée. L'utilisation de Python dans le logiciel comble cinq besoins clés :

- servir à construire l'interface graphique de Modeleur2;
- devenir le remplaçant du langage de la calculatrice de Modeleur 1.0;
- agir comme pièce centrale du système de plug-in (ouverture vers l'extérieur);
- servir de moyen pour tester le logiciel par envoi d'événements;
- permettre d'enregistrer et de rejouer les événements sur le bus (macros).

Effectuer le passage vers une base de données relationnelle

Modeleur2 est construit autour d'une base de données relationnelle. Ceci donne accès aux fonctionnalités pour les requêtes spatiales, en plus de valider intrinsèquement la cohérence entre les données.

Utiliser VTK et MayaVi pour les représentations graphiques

Modeleur2 base ses fonctionnalités graphiques sur VTK (Visual ToolKit), une librairie graphique relativement de haut niveau. Modeleur2 intègre également le composant MayaVi qui, pour sa part, rend disponible une gamme étendue de fonctionnalités de représentation graphique et donne accès à tous les paramètres VTK.

Mettre en place automatisé de validation du logiciel

Un système de validation automatisé a été mis en place pour assurer une base de code fonctionnelle et validée en tout temps. Le code a donc été construit sur de multiples

compilateurs et répond au standard C++. Dans la même veine, un banc de test a été conçu et tous les scénarios de tests sont exécutés quotidiennement assurant la fonctionnalité de chacun des éléments.

6.2.3 État des logiciels

6.2.3.1 Modeleur2

Les plug-in suivants sont intégrés à Modeleur2 et offrent la plupart des fonctionnalités qu'ils se devaient d'offrir (voir la section 4.1.4 qui discute du bilan détaillé) :

- Projets et gestion des données;
- Validation des données;
- MNT;
- Partition de maillage;
- Assemblage des données sur un maillage;
- Affichage;
- Visualisation;
- Construction de séries;
- Sonde.

Par manque de temps et de ressources, certains aspects du logiciel n'ont pu être abordés. Les partitions de conditions aux limites et la publication des résultats en sont un exemple.

De plus, certaines fonctionnalités sont incomplètes, comme c'est le cas des activités de validation des données et de construction du MNT qui ne supportent que les données de topographie. Le module de simulation, pour sa part, bien qu'il soit relativement avancé, n'a pu être finalisé et il n'est donc pas possible d'exécuter des simulations avec la version livrée.

Les outils d'analyse suivants, tous mentionnés dans les spécifications fonctionnelles, n'ont pu être implantés : outil de lignes, outil de surfaces, outil de vue tabulée, outil de projection (d'un champ éléments finis vers un autre maillage) et outil de projection cartographique.

Au cours du projet, priorisant la mise en place des fonctionnalités clé, peu de la force de travail a été concentrée sur l'aspect esthétique. Il reste du travail à faire sur cet aspect.

Enfin, il y a également un travail à faire pour optimiser l'utilisation de la base de données afin de pouvoir travailler avec des cas réels, plutôt qu'avec des cas simples.

6.2.3.2 H2D2

H2D2 est fonctionnel et a été intégré à Modeleur1. H2D2 a été utilisé pour des calculs instationnaires de transport-diffusion où la séparation des propriétés nodales sur des fichiers indépendants et la résolution parallélisée par LDU sur disque ont permis de résoudre de gros problèmes.

L'élément de St-Venant est programmé mais n'est pas encore opérationnel. Une comparaison serrée entre Hydrosim et H2D2 reste à faire, tant au niveau des taux de convergences que des résultats.

Au niveau du calcul réparti, si la programmation est achevée, il reste ici également à valider et tester cette partie pour déterminer si les gains attendus en terme de vitesse de calcul se réalisent. Pour sa part, le nouvel élément 2D ½ est en cours d'implantation.

Bien que le logiciel ne soit pas entièrement complété, nous possédons désormais en H2D2, une plateforme avec laquelle il est beaucoup plus facile de travailler et à laquelle il sera aisé d'ajouter des fonctionnalités.

6.3 Bilan global

Au niveau scientifique, nous avons pu, au cours de ce projet, développer et valider une méthodologie pour l'analyse et la réduction des risques d'inondation. Cette méthodologie s'exprime en huit étapes :

1. Tenir compte de la portée réelle des coûts du risque
2. Construire une base de données sur les aléas de crue et d'embâcles
3. Établir la probabilité actuelle et évolutive des aléas
4. Simuler les conditions hydrauliques d'inondation actuelles et potentielles
5. Regrouper les données disponibles sur la vulnérabilité des constructions
6. Établir la distribution des dommages et du risque
7. Rechercher des solutions et évaluer leur efficacité technique
8. Proposer une stratégie intégrée

Si l'approche décrite semble naturelle et relativement classique, son application intégrale compte très peu sinon pas de cas similaire dans les études antérieures du moins à l'échelle du Québec. L'étude de Châteauguay (Leclerc et al., 2006) a poussé très loin toutes les dimensions innovatrices de l'approche.

Au niveau logiciel, beaucoup de travail de fond a été réalisé pendant ces trois années. Le logiciel est construit de façon à pouvoir être extensible dans l'avenir. Tout est en place pour amener progressivement le logiciel à maturité en développant autour du noyau qui a été construit ces dernières années.

La méthodologie de développement axée sur des stagiaires nous a permis d'accomplir une grande quantité de travail avec des ressources relativement limitées. Toutefois, cette stratégie a aussi démontré ses limites. On s'est rendu compte que le développement était à la merci des compétences des étudiants que nous étions en mesure de recruter. Les sujets traités lors des stages étaient excessivement spécialisés et il était souvent difficile de trouver la main d'œuvre appropriée.

*Dans le même ordre d'idée, il est extrêmement ardu de recruter des étudiants à la maîtrise qualifiés pour travailler sur ce type de projet. Les candidats qui allient des aptitudes dans le domaine des éléments finis, de l'hydraulique fluviale, de la modélisation numérique et de la biologie sont peu nombreux et difficiles à recruter étant donné le niveau de difficulté élevé de ce champ d'étude.

Malgré tout, nous sommes fiers d'avoir pu offrir des stages si riches et si variés, tout en menant à terme un projet logiciel de cette envergure.

Sans l'ombre d'un doute, cette méthodologie de développement axée sur des stagiaires a des retombées fort positives pour le Québec de demain. Non seulement, un projet comme le projet SEGRI a pour but d'aider à résoudre des problématiques qui touchent de près la population, mais en plus il contribue à la formation de personnel hautement qualifié.

7 Bibliographie

Sites web

Boa Constructor : http://boa-constructor.sourceforge.net/. Cité le 5 janvier 2006.

Boost : http://www.boost.org/. Cité le 5 janvier 2006.

GEOS : http://geos.refractions.net. Cité le 5 janvier 2006.

MayaVi : http://mayavi.sourceforge.net/ . Cité le 5 janvier 2006.

MPICH2 : http://www-unix.mcs.anl.gov/mpi/mpich2. Cité le 5 janvier 2006.

OpenMP : http://www.openmp.org. Cité le 5 janvier 2006.

OTL : http://otl.sourceforge.net/. Cité le 5 janvier 2006.

ParMETIS : http://www-users.cs.umn.edu/~karypis/metis/index.html. Cité le 5 janvier 2006.

PETSc : http://www-unix.mcs.anl.gov/petsc/petsc-2/index.html. Cité le 5 janvier 2006.

PostgreSQL : http://www.postgresql.org. Cité le 5 janvier 2006.

PostGIS : http://www.postgis.org. Cité le 5 janvier 2006.

Python : http://www.python.org. Cité le 5 janvier 2006

SuperLU : http://www.cs.berkeley.edu/~demmel/SuperLU.html. Cité le 5 janvier 2006.

VTK : http://www.vtk.org. Cité le 5 janvier 2006.

Winpdb : http://www.digitalpeers.com/pythondebugger/. Cité le 5 janvier 2006.

WxPython : http://www.wxpython.org. Cité le 5 janvier 2006.

Documents papier

Bonnifait, L. et Leclerc M. (2004). Construction de courbes niveau-dommages pour l'habitat québécois. Pour le compte d'Environnement Canada. Rapport INRS-ETE #R-728. Mars, 15 pages. Février.

Bonnifait, L. (2005). Développement de courbes submersion-dommages pour l'habitat résidentiel québécois. Mémoire de maîtrise de l'INRS-ETE. Décembre. 87 p.

Blin, P., Leclerc M., Secretan Y. et Morse B. (2005) Cartographie du risque unitaire d'endommagement (CRUE) par inondations pour les résidences unifamiliales du Québec. Revue des Sciences de l'Eau, 18(5). Décembre.

Doyon, B., Côté, J.-P., Bonnifait, L., Roy, N., Morin, A. et Dallaire, É. (2004). Crues du fleuve Saint-Laurent : construction de courbes d'endommagement par submersion applicables aux résidences installées dans la plaine inondable. Rapport technique SMC Québec – Section Hydrologie RT-132, Environnement Canada, Ste-Foy, 51 pages.

Heniche, Y. Secretan et M. Leclerc. "HYDROSIM 1.0a06 : Guide d'utilisation". Rapport INRS-Eau R482-G2F, 2000.

Heniche M., Y. Secretan et M. Leclerc. "DISPERSIM 1.0a02 : Guide d'utilisation". Pour Environnement Canada. Rapport INRS-Eau R558-G2, 2001.

Le Boulleur de Courlon, R. (2003). Application du projet SEGRI : étude des risques d'inondation de la rivière Chaudière. Rapport de stage INRS-ETE à l'ENSEEIHT (Toulouse).

Leclerc, M., Secretan Y., Heniche M., Ouarda, T.B.M.J. et Marion J. (2003). Une méthode prédictive non biaisée et géoréférencée d'estimation des dommages résidentiels d'inondation. Revue canadienne de Génie civil. 30(5) : 914-922.

Leclerc, M., Morse B. et Secretan Y. (2005). Estimating the flood risk related to ice jams. Soumis à Cold Reg. Sc. & Tech., Décembre.

Leclerc, M., Boudreau, P., Roy N., Secretan Y., El Adlouni, S., Ouarda T., Chaumont D., Falardeau I. et Morneau F. (2006). Contribution à la recherche d'une solution intégrée au risque d'inondation à Châteauguay. Pour le compte de la Ville de Châteauguay, en collaboration avec le ministère de la Sécurité publique. Décembre.

Secretan Y., M. Heniche et M. Leclerc (2000). Dispersim : un outil de modélisation par éléments finis de la dispersion de contaminants du milieu fluvial. Travail réalisé pour le compte de Pêches et Océans Canada. Garde Côtière. Rapport scientifique INRS-Eau No. R558, 57 pages.

Secretan, Y., Larouche, E. (2003). Système d'Évaluation et de Gestion des Risques d'Inondation en milieu fluvial (SEGRI) : Rapport d'étape #1. Québec, INRS-Eau, Terre & Environnement. 41 pages. (INRS-Eau, Terre & Environnement, rapport de recherche 720 e1).

Secretan, Y., Larouche, E., le Boulleur de Courlon, R., Ouellet, S., Derenne, M., Solinski, K., Kaczor, O. et Larrivée, F. (2003). Système d'évaluation et de gestion des risques

d'inondation en milieu fluvial (SEGRI). Rapport de recherche. Québec, INRS-Eau, Terre & Environnement. Pagination multiple. (INRS-Eau, Terre & Environnement, rapport de recherche; 720 a).

Secretan, Y., Leclerc, M., Larouche, É. et Boudreau, P. (2003). Système d'évaluation et de gestion des risques d'inondation en milieu fluvial (SEGRI) : spécifications – Modeleur2. Québec, INRS-Eau, Terre & Environnement, viii, 60 pages. (INRS-Eau, Terre & Environnement, rapport de recherche; 720 b1).

Secretan, Y., Larouche, E. (2004). Système d'Évaluation et de Gestion des Risques d'Inondation en milieu fluvial (SEGRI) : Rapport d'étape #2. Québec, INRS-Eau, Terre & Environnement. 45 pages. (INRS-Eau, Terre & Environnement, rapport de recherche 720 e2).

Secretan, Y., Larouche, E., Giasson, M., Caron, C., Behaghel, B., Nadeau, D., Lévesque, S. et Labbé, S. (2004). Système d'évaluation et de gestion des risques d'inondation en milieu fluvial (SEGRI) : Rapports de recherche 2004. Québec, INRS-Eau, Terre & Environnement. Pagination multiple. (INRS-Eau, Terre & Environnement, rapport de recherche 720 b).

Secretan, Y., Leclerc, M., Larouche, É. et Boudreau, P. (2004). Système d'évaluation et de gestion des risques d'inondation en milieu fluvial (SEGRI) : spécifications – Modeleur2. Québec, INRS-Eau, Terre & Environnement. 126 pages. (INRS-Eau, Terre & Environnement, rapport de recherche 720 b2).

8 Glossaire

API :

Application Programming Interface : Interface de programmation d'applications. Ensemble de fonctions qui répondent à une interface. Le programmeur utilise l'API sans avoir à se soucier de l'implémentation sous-jacente.

Arborescence :

Représentation organisationnelle selon une structure arborescente, qui établit la stricte hiérarchie entre les divers composants, de façon que toute information, sauf la première, procède d'une seule autre, mais peut en engendrer plusieurs.

Base de données relationnelle :

Base de données organisée en fonction des relations qui existent entre les données. Le principal langage utilisé dans les bases de données relationnelles est le SQL.

Bus d'événements:

Mécanisme propriétaire à Modeleur2 ayant pour but d'effectuer la distribution d'événements aux différents modules qui y sont reliés. Le bus d'événements contient une liste des modules et distribue les événements à chaque module dans cette liste.

Calcul:

Composante de base d'une simulation dans Modeleur2. Chaque calcul est réalisé à l'aide de paramètres d'entrée et produit une série de résultats (ex : champ de niveaux d'eau et champs de vitesses). Les résultats d'un calcul peuvent à leur tour servir de paramètres d'entrée (i.e. solution initiale) à un autre calcul.

Callback :

Dans le contexte de la programmation informatique, procédure appelée lorsque certains événements préalablement définis se produisent.

Champ :

Dans Modeleur2, un champ est une structure de données qui possède en plus des propriétés algébriques de type algèbre, des propriétés spatiales. Le champ porte des données qui répondent elles aussi aux structures algébriques et sont associées à une position dans l'espace. Il est possible d'interroger le champ au sujet de sa valeur en tout point compris dans la région spatiale qu'il englobe. Sous Modeleur2, les champs peuvent être qualifiés par la valeur qu'il portent (ex : valeur scalaire, vectorielle ou ellipse d'erreur) et peuvent être qualifiés par la méthode qu'ils utilisent pour arriver à fournir la valeur en tout point (ex : éléments finis, analytique, etc).

Champ analytique :

Le champ analytique est un champ défini par une équation mathématique. (ex : $f(x,y) = 2x + y$).

Champs éléments finis :

Le champ éléments finis est la structure de données rassemblant les valeurs nodales sur un maillage éléments finis. Une valeur en tout point de la région spatiale du champ est obtenue par interpolation d'éléments finis.

Champ d'ellipse erreur :

Type de champ pour lequel la donnée portée correspond aux paramètres d'une ellipse décrivant l'erreur à ce point.

Champ publié :

Dans Modeleur2, le terme champ publié fait référence à un semi de points qui a été validé par l'usager, et que l'on ensuite transformé en champ en générant un maillage avec les points du semi validé.

Champ scalaire :

Type de champ pour lequel la donnée portée correspond à une valeur réelle.

Champ vectoriel :

Type de champ pour lequel la donnée portée correspond à un vecteur.

Compiler :

Traduire le code source d'une application écrite dans un langage de haut niveau en code objet formé d'instructions du langage machine.

Condition (d'une partition de conditions aux limites) :

Une condition d'une partition de conditions aux limites correspond à une valeur de paramètre (ex : niveau d'eau, débit, etc) qui doit prévaloir à cette limite du domaine. Les conditions dépendent du simulateur utilisé.

Connexion ODBC :

Opération qui permet d'établir une liaison entre une application et une base de données via une interface ODBC.

Condition aux limites :

Dans Modeleur2, le terme « condition aux limites » correspond au résultat d'une opération d'assemblage d'une partition de conditions aux limites sur un maillage. Il est bien important de distinguer ce terme de celui de condition qui lui est une composante d'une partition de conditions aux limites (voir Condition).

Couche :

Dans Modeleur2, une couche est une construction géométrique constituée d'un ou plusieurs sous-domaines fermés. Chaque couche est associée à un jeu de données

ou à des paramètres. Chaque couche se voit assigner un ordre de priorité. qui permet de déterminer de quelle couche le jeu de donnée ou les paramètres doivent être extraits lorsque deux ou plusieurs couches se chevauchent dans une partition.

Couche de maillage:

Couche appartenant à une partition de maillage. La donnée portée par la couche correspond aux paramètres de maillage utilisés pour mailler.

Couche de données d'un MNT (ex : couche de topographie):

Couche appartenant à une partition dans un MNT (ex : partition de topographie). La donnée portée par la couche correspond au champ qui décrit la couche.

C++ :

Langage de programmation de haut niveau défini comme une version orientée objet du langage C. Le C++ est le principal langage de programmation utilisé dans l'écriture du code source de Modeleur2.

Degré de raffinement d'un maillage :

Correspond à la grosseur des mailles d'un maillage. Un haut niveau de raffinement implique des mailles plus petites, et un faible degré de raffinement, des mailles plus grosses.

DLL :

Dynamic Linked Library (bibliothèque de liens dynamiques). Ensemble de routines qui sont disponibles aux applications au moment de l'exécution.

Échelle de précision :

L'échelle de précision est une valeur qui qualifie l'erreur générée par le retrait d'un point. Par exemple, dans un semi de points de topographie, un terrain de soccer parfait peut être décrit par quatre points. Tous les autres points peuvent avoir un échelle de précision inférieure à ces quatre points car leur retrait n'enlève rien à la description du terrain.

Espace de représentation :

Dans Modeleur2, un espace de représentation correspond au contenu d'une fenêtre graphique. Un espace de représentation est composé de trois plans : plan arrière, plan médian et plan interactif.

Événement :

Dans Modeleur2, ceci correspond à l'unité d'échange sur le bus d'événements. Un événement peut contenir à la fois de l'information, une commande ou encore rien.

Fenêtre graphique :

Fenêtre dans laquelle il y peut y avoir affichage de représentations graphiques.

Filtrage :

Dans Modeleur2, le terme filtrage fait référence à l'opération qui permet distinguer les points d'un semi qui doivent être conservés de ceux qui doivent être désactivés. L'opération de filtrage permet donc de nettoyer le semi brut des points indésirables qu'il peut contenir.

Filtre primaire :

Les filtres primaires sont utilisés pour calculer une cote de qualité à chaque point du semi introduit à l'entrée du filtre. Ces filtres peuvent baser leur appréciation sur différents paramètres, et ils attribuent à chaque point du semi une cote entre 0 et 1.

Filtre secondaire :

Les filtres secondaires sont les filtres qui travaillent à partir des cotes produites par les filtres primaires. Ils ont comme rôle d'indiquer si un point doit être gardé ou retiré en se basant sur les cotes combinées des filtres primaires et d'une valeur seuil.

Focus :

Zone d'une fenêtre active où l'utilisateur fera sa prochaine intervention. Ceci coïncide avec la position du curseur, après que l'utilisateur a cliqué sur la souris.

GIS :

Geographic Information System. Voir SIG.

GUI:

Graphical User Interface (interface utilisateur graphique). Interface utilisateur basée sur l'emploi d'éléments graphiques tels que les fenêtres, les icônes et les menus, qui visent la simplicité d'emploi et qui créent un environnement de travail convivial.

IDE :

Integrated Development Environment (environnement de développement intégré). Dans un système de développement, ensemble d'outils intégrés qui sont utilisés pour le développement de logiciels et qui sont directement accessibles à partir de l'interface utilisateur.

Interpolation :

Estimation ou calcul de valeurs intermédiaires dans une série de valeurs connues.

Interpréteur :

Programme qui traduit les instructions d'un langage évolué en langage machine et les exécute au fur et à mesure qu'elles se présentent.

Isolignes :

Ligne qui joint des points correspondants à une même valeur. Comme exemple d'isolignes, on peut mentionner les courbes de niveau.

Isosurfaces :

Surface qui englobe des points correspondant tous à une même plage de valeurs.

Jeu de données :

Dans Modeleur2, le terme jeu de données fait référence aux champs publiés à l'étape de validation, et que l'usager doit spécifier lors de la construction d'un MNT.

Langage interprété (langage de scripts) :

Langage de programmation évolué qui traduit en langage machine le code source d'un programme (d'un script), pour l'exécuter immédiatement, et ce, ligne par ligne.

Liens externes :

Dans Modeleur2, un lien externe correspond à un lien établi avec une donnée appartenant à un projet autre que le projet courant. Par exemple, le projet 'a' peut avoir comme lien externe, le maillage du projet 'b'.

Link :

Séquence d'instructions reliant les différents modules ou les sous-programmes qui sont nécessaires à l'exécution d'une fonctionnalité particulière, de façon à créer un programme cohérent et exécutable.

Liste d'affichage :

Représentation sous forme d'une arborescence du contenu de chacune des fenêtres graphiques, et pour chacune d'elles du contenu de ses plans, et des objets graphiques contenus par ceux-ci.

Log :

Relevé chronologique des opérations informatiques constituant un historique de l'utilisation. Dans Modeleur2, le log (journal) fournit à l'usager de l'information sur le déroulement des opérations exécutées. Le log peut être configuré pour afficher différents niveaux de détail.

Macro :

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. Dans Modeleur2, une macro correspond en fait à une suite d'événements enregistrés dans un script Python.

Maillage:
Surface ou volume généré par un ensemble de nœuds et de lignes de contrôle dont les caractéristiques sont définies par les points ou les courbes qui ont servi à la création de l'élément.

Mailleur :
Dans Modeleur2, le terme mailleur correspond à la partie de logiciel utilisée pour générer un maillage (i.e. générer les nœuds et les éléments).

Métadonnées :
Donnée qui renseigne sur la nature de certaines autres données et qui permet ainsi leur utilisation pertinente.

Méthode des éléments finis :
La méthode des éléments finis est une technique d'interpolation qui consiste à discrétiser le domaine de calcul en un ensemble de sous domaines de géométrie simple, les éléments. Sur chaque élément, on définit une fonction d'interpolation qui prend en compte la disposition des nœuds de l'élément (géométrie) et les valeurs qu'ils portent.

MNT :
Modèle Numérique de Terrain. Dans Modeleur2, le MNT correspond à un ensemble de partitions qui décrivent le terrain (ex : partition de topographie, partition de substrat, etc).

Module C++ :
Partie C++ d'un plug-in Modeleur2 connectée au bus d'événements afin de pouvoir y envoyer des événements et en recevoir.

Module Python :
Un module Python est un élément qui fournit de nouvelles fonctions à un interpréteur Python. Dans Modeleur2, les scripts, les macros et les DLL Python (.pyd) sont considérés comme des modules Python.

Multipolygone :
Ensemble de polygones.

Multipolyligne :
Ensemble de polylignes.

Niveau de tolérance à l'erreur :
Dans Modeleur2, le niveau de tolérance à l'erreur correspond au niveau d'erreur acceptable pour lequel une couche est considérée maillée avec un degré de raffinement acceptable.

Objet graphique :

Dans Modeleur2, toute représentation affichée dans une fenêtre graphique est considérée comme un objet graphique. Par exemple, le contour d'une couche est un objet graphique de même que la représentation graphique d'un maillage.

ODBC :

Interface standardisée, définie par Microsoft, permettant d'accéder à des bases de données ayant des formats différents.

Onglet :

Partie qui semble dépasser du bord d'un élément affiché dans une fenêtre et qui sert à accéder facilement à des fonctions, des réglages, des rubriques d'aide, des dossiers ou des documents, en cliquant dessus avec un dispositif de pointage tel que la souris.

Open source :

Code source que l'on rend disponible gratuitement pour qu'il puisse être modifié et redistribué, dans un contexte de développement communautaire.

OTL :

Librairie qui permet d'effectuer la communication avec la base de données via les opérateur de flot (>> et <<) pour lire de la base de données et écrire dans la base de données) en encapsulant le lien ODBC.

Partition :

Sous Modeleur2, le terme partition fait référence à un ensemble de couches.

Partition de conditions aux limites :

Type de partition qui décrit les conditions aux limites d'un domaine. Ce type de partition sera utilisé pour procéder à l'assemblage des conditions aux limites sur un maillage.

Partition générique :

Type de partition qui portent un type de donnée qui a été défini par l'utilisateur. La partition générique peut supporter l'interpolation ou non.

Partition de maillage :

Type de partition dont les couches servent à générer un maillage.

Partition spécialisée :

Type de partition prédéfini dans Modeleur2 (topographie, substrat, etc) par opposition avec les partitions générique qui elle sont définies par l'utilisateur.

Plug-in :

Extension à une application qui vient se loger dans l'application elle-même. Une fois installé, on peut avoir accès aux fonctionnalités du plug-in de façon tout à fait transparente.

Polygone :

Figure formée par une suite ordonnée de segments (côtés), dont chacun a une extrémité commune (sommet) avec le précédent et le suivant.

Polylignes :

Entité géométrique constituée d'une succession de lignes parfaitement reliées les unes aux autres par leurs extrémités.

Projection :

Transposition d'une portion de l'ellipsoïde de référence géodésique représentant la surface terrestre, sur une surface plane, à l'aide d'un modèle mathématique.

Projet :

Dans Modeleur2, le projet est le contenant qui héberge toutes les données et résultat de l'utilisateur dans le cadre d'une étude.

Python :

Un langage de programmation interprété, interactif, orienté-objet comparable à Tcl, Perl ou Java.

RAD :

Rapid Application Development. Développement rapide d'application avec des outils modernes.

Région spatiale:

Région définie par des coordonnées {x,y,z} de l'espace géographique.

Représentation graphique :

Dans Modeleur2, le terme représentation graphique fait référence au contenu d'une fenêtre graphique. Tout affichage dans une fenêtre graphique est une représentation graphique (ex : affichage d'un maillage, affichage d'une partition, etc.).

Schéma :

Dans un système de gestion de base de données (SGBD), description d'une base de données créée au moyen du langage de définition de données proposé par le SGBD. Un schéma permet de regrouper un ensemble de tables. Dans Modeleur2, chaque projet est associé à un schéma.

Script :

Série d'instructions servant à accomplir une tâche particulière.

Semi de points :

Sous Modeleur2, le terme semi de points correspond à un ensembles de points {xy, valeur}. À titre d'exemple, pour un semi de points de topographie, la valeur correspond à l'altitude (z).

Série :

Une série peut être vue comme un champ éléments finis, mais pour lequel le domaine n'est pas une région spatiale mais plutôt une région d'un espace conceptuel de solution. Tout comme le maillage, la série possède des nœuds et des éléments, lesquels sont utilisés pour interpoler à partir de valeurs portées par les nœuds. L'emplacement de chacun de nœuds est spécifié via une coordonnée. Typiquement, la coordonnée d'une série est entière ou réelle. Une série peut être en 1D et 2D, et théoriquement, pourrait même être en n dimensions. Les nœuds d'une série portent des valeurs. Les valeurs peuvent être simples comme des scalaires (ex : niveau d'eau), des vecteurs (ex : vitesse) mais peuvent être également des champs répartis spatialement. Les données d'une série doivent être en relation.

SGBD :

Système matériel et logiciel dont la fonction est d'assurer la gestion automatique d'une base de données et de permettre la création, la modification, l'utilisation et la protection des données.

SIG :

Système d'information portant sur des données géographiques.

Simulateur :

Dans Modeleur2, le terme simulateur fait référence au logiciel utilisé pour effectuer un calcul, en fonction de paramètres d'entrée.

Simulation :

Une simulation est une suite d'étapes de calcul (ex : hydrodynamique) qui a pour but d'obtenir une solution finale satisfaisante

Sous-domaine :

Polygone géoréférencé qui définit une couche ou une partie d'une couche.

Squelette du MNT :

Correspond à la plus petite zone commune à un ensemble de partitions spécifié par l'usager. Il peut servir lors de la construction d'un maillage pour s'assurer que le maillage ne dépasse pas la zone de données partagée par toutes les partitions d'intérêt.

SQL :

Langage d'interrogation, de mise à jour et de gestion des bases de données relationnelles.

Table :

Mode d'organisation des données dans une base de données, composé de colonnes et de lignes, dont chaque cellule contient des informations et des liens qui existent entre les données. La table constitue la structure de base de la relation entre les données.

Trigger :

Mécanisme qui amène l'exécution automatique d'une procédure lorsqu'un utilisateur s'apprête à modifier le contenu d'une base de données. En mettant en marche une procédure déterminée lorsqu'un événement suspect se présente, le déclencheur assure le maintien de l'intégrité des données dans la base de données.

VTK :

Visual ToolKit. Librairie graphique qui offre des fonctionnalités haut niveau. Cette librairie est utilisée dans Modeleur2 pour effectuer toute tâche de représentation graphique.

Vue BD :

Dans les bases de données relationnelles, table logique créée par la spécification d'opérations relationnelles reliant plusieurs fichiers. Dans de nombreux systèmes, une vue peut être manipulée comme s'il s'agissait d'une table réelle.

Zone d'affichage :

Dans Modeleur2, la zone d'affichage correspond à la zone délimitée par une fenêtre graphique. Plus l'utilisateur zoom in, plus la zone d'affichage est réduite, et inversement, plus il zoom out, plus elle est agrandie.

Zone de contrôle :

Dans Modeleur2, la zone de contrôle correspond à la partie de l'interface utilisateur où l'usager peut gérer ses données. Cette zone contient entre autres les arborescences des différents types de données, mais pourrait également accueillir tout type de contrôle offert par un plug-in.

Zone d'étude :

Zone (ou région) d'intérêt sur laquelle l'usager désire porter son étude. Cette zone d'étude limitera les données chargées lors des requêtes de chargement.

Sources de certaines définitions : http://www.grandictionnaire.com