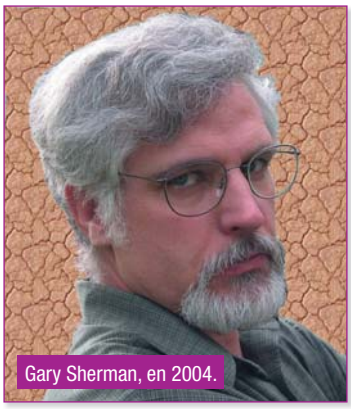
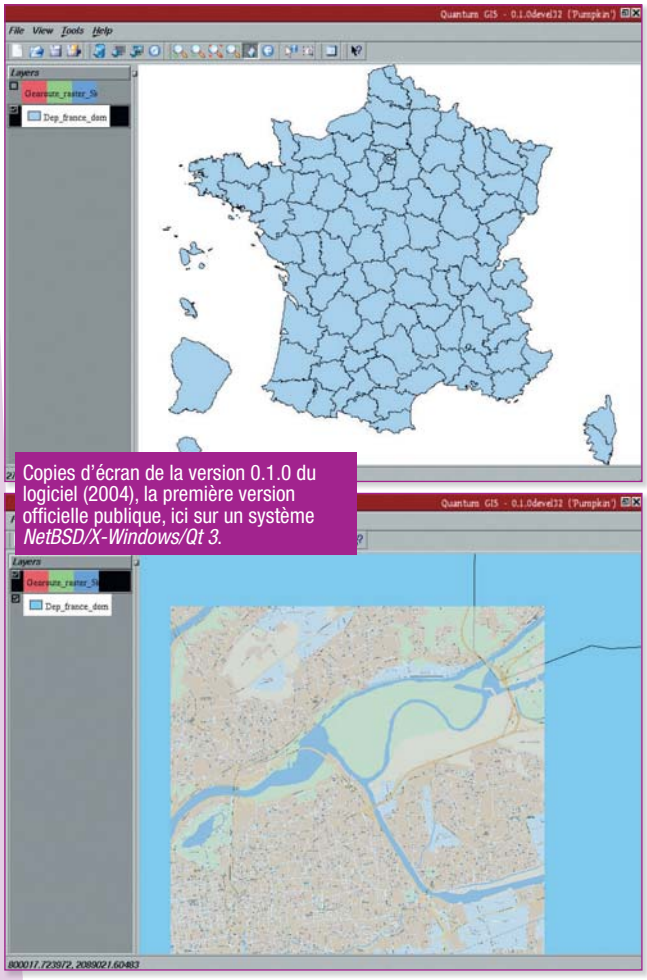
QGis 2 : Révolution ou évolution ?

La sortie de la version 2.0, très attendue, du logiciel de SIG libre le plus populaire marque l'arrivée d'un grand nombre d'évolutions. S'agit-il pour autant d'une révolution, et que peut-on faire avec la nouvelle plate-forme. Réponse au travers du témoignage croisé de deux utilisateurs.

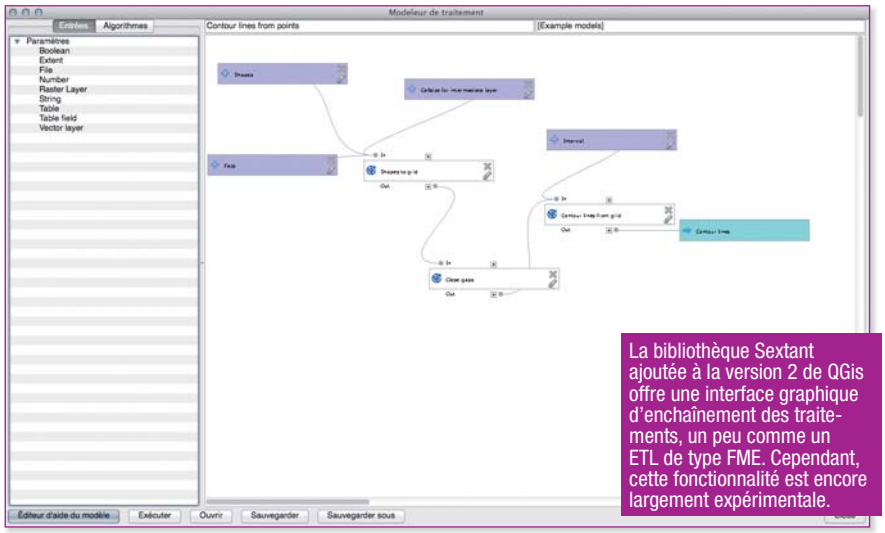
Qui ne connaît pas *QGis* ? On peut, sans grand risque, avancer que ce logiciel (l'appellation *Quantum GIS* étant officiellement délaissée au profit du simple *QGis*) est devenu le SIG *Open Source* le plus populaire dans le monde. Osons une petite rétrospective. En avril 2004 - il y aura donc bientôt dix ans -, à l'occasion du numéro 34 de *Géomatique Expert*, Gary Sherman, le fondateur et leader, pendant des années, du projet QGis (Gary Sherman joue désormais plus un rôle de « *président honoraire* » que de contributeur actif au code) s'exprimait sur la version 0.1.0 (!). À l'époque, il nous expliquait que « *QGis se voulait un programme rapide et avant tout ergonomique pour afficher des données géographiques […] Le principe modulaire, un noyau sur lequel se greffent des extensions, va nous permettre d'ajouter, au fur et à mesure, de nouvelles fonctions et des capacités de géotraitement.* » Interrogé sur sa vision d'une version idéale de *QGis*, Gary Sherman répondait : « *Idéalement, je voudrais que QGis possède au moins un éditeur d'objets vectoriels, de labels, la possibilité d'ajouter des annotations, des capacités de production cartographique, un calculateur de transformations/projections et d'autres fonctions que l'on trouve couramment dans les logiciels commerciaux* ». À quoi ressemblerait un SIG libre « *parfait* » ? « *Ce serait un logiciel facile à installer, encore plus facile à utiliser, et qui offrirait des fonctions à la fois simples et avancées, en particulier l'édition.* » Enfin, Gary Sherman espérait que « *le logiciel se fera connaître, reconnaître et sera utilisé par un grand nombre de géomaticiens* ». Dix ans après, le pari semble avoir été tenu, et les objectifs plus que remplis.


Gary Sherman, en 2004.


Copies d'écran de la version 0.1.0 du logiciel (2004), la première version officielle publique, ici sur un système *NetBSD/X-Windows/Qt 3*.

Avec la version 2.0 de l'automne dernier, on peut même dire que le logiciel a dépassé les limites du SIG au sens strict pour s'aventurer vers la plate-forme fédératrice. La philosophie initiale du produit, à savoir en faire un produit modulaire capable, pour s'étendre, de s'interfacer facilement avec d'autres logiciels ou bibliothèques libres développées par ailleurs, assume désormais tout son sens. Cette polyvalence s'exprime dans l'ajout de la bibliothèque *Sextant*, écrite en *Python*. Cette

dernière assure la jonction entre l'environnement de visualisation propre à *QGis* et une batterie de bibliothèques d'algorithmes spécialisés, certains issus de logiciels SIG libres tiers, comme *Grass* ou *Saga* (un produit récent d'une université allemande), la bibliothèque ultra-spécialisée OTB (*Orfeo Tool Box*), sur laquelle nous reviendrons, et… les algorithmes de traitement de *QGis* lui-même. « *L'idée*, explique Florian Boret, géomaticien chez *Alisé Géomatique, c'est que la bibliothèque* Sextant *fournisse un outil de chaînage graphique, à la* Model builder™*, pour réaliser des traitements complexes, y compris avec les fonctions de géotraitement incluses dans QGis lui-même. À l'heure actuelle, cet outil existe, mais ne peut guère exécuter correctement que quelques tâches très simples. Mais je suis confiant sur l'avenir de ce module.* »



La bibliothèque Sextant ajoutée à la version 2 de QGis offre une interface graphique d'enchaînement des traitements, un peu comme un ETL de type FME. Cependant, cette fonctionnalité est encore largement expérimentale.

La bibliothèque Sextant

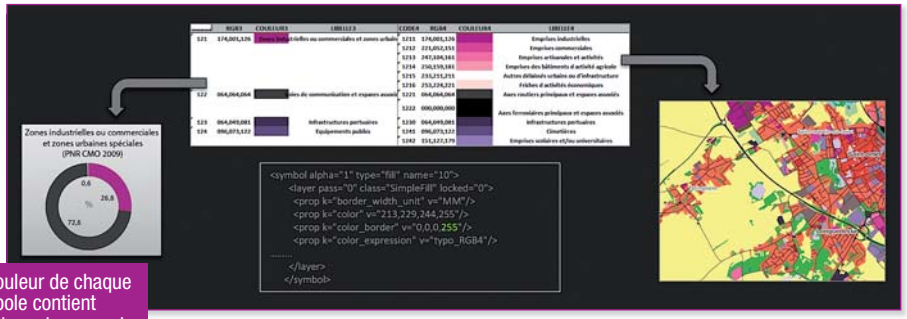
Les ressources mises à disposition de l'utilisateur de *QGis* par l'intermédiaire de la bibliothèque *Sextant* en font l'un des SIG les plus puissants en termes de possibilité d'analyse. « *La bibliothèque Orfeo*, poursuit Florian Boret, *est issue d'un développement financé par le CNES, puis récupéré en interne par ses soins. C'est une boîte à outil très complexe, incluant des algorithmes jusqu'ici réservés à des logiciels très onéreux, comme la segmentation/classification orientée objet. L'arrivée de ce type de traitement dans une bibliothèque* Open source *marque une double avancée : la première, c'est évidemment sa disponibilité pour tous, quel que soit son budget ; l'autre volet, c'est l'ouverture du code. Jusqu'ici, l'approche orientée objet était considérée comme un secret précieusement gardé, avec peu, voire pas du tout d'informations sur son fonctionnement : il fallait plus ou moins deviner à quoi servaient les options. Désormais, tout le monde peut en comprendre les principes de fonctionnement et s'en servir. J'espère que ce sera l'occasion de démocratiser cette approche extrêmement puissante, mais dont le coût entravait jusqu'ici la diffusion. Or, nous manquons précisément de retour d'expérience pour pouvoir nous servir des ces algorithmes efficacement.*

*L'*Orfeotoolbox *est donc extrêmement puissante, mais elle requiert un haut niveau d'expertise. Par ailleurs, pour les experts, l'utilisation via la bibliothèque* Sextant *risque de s'avérer quelque peu décevante, car toutes les options disponibles en ligne de commande ne sont pas présentes dans la version GUI, donc on ne peut pas porter des scripts complexes, comme ceux développés à l'ONF, par exemple. Malgré tout, les opérations standard, comme la classification supervisée, fonctionnent parfaitement, et l'ergonomie est intéressante.* »

La bibliothèque *Sextant* offre également les algorithmes plus traditionnels issus des logiciels *Grass*, *Saga* ainsi que de la bibliothèque *GDAL*, aussi bien dans le domaine de l'hydrologie que dans celui des analyses *raster* pour les calculs de pente ou de bassins versants, par exemple. Toutes ces possibilités additionnelles font de *QGis* une plate-forme particulièrement polyvalente. En revanche, cela signifie également que le logiciel enfle à chaque nouvelle version. On est bien loin du visualisateur « *taille de guêpe* » des premiers temps. Sous *Unix*, *Linux* ou *OS X*, pour peu que l'on opte pour une installation au travers d'un gestionnaire de packetage, on peut limiter le nombre de bibliothèques tierces au strict nécessaire, voire ne pas en installer du tout. À l'inverse, la plupart des installations binaires, y compris sous *Windows*™, ont choisi une approche maximaliste ; en pratique, installer *QGis* 2 revient à installer *QGis*, *GDAL*, *Grass*, *Saga* et l'*Orfeo Toolbox*.

Symbologie

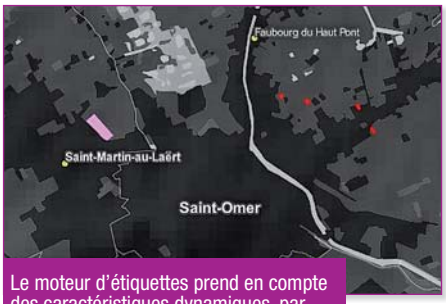
L'autre domaine dans lequel QGis 2 progresse significativement par rapport à la version 1.8 est la symbologie. Le moteur a



La couleur de chaque symbole contient maintenant une quatrième composante (alpha), codant sa transparence. On peut donc l'ajuster symbole par symbole, et non plus uniquement globalement par couche (illustration Alisé Géomatique).

été entièrement revu, il est non seulement plus puissant, mais il offre de nombreuses possibilités supplémentaires totalement inédites. La première innovation consiste à pouvoir sauvegarder le style avec la couche. « *Un petit détail, mais qui peut avoir de l'importance*, indique Hélène Durand, gérante d'Alisé Géomatique. *Quand, dans un projet multi-partenaires, vous discutez, rediscutez, re-rediscutez… de la symbologie, des nuances de couleurs, de l'épaisseur des bordures d'une couche, vous êtes particulièrement content de pouvoir graver ces choix dans la table, de sorte à ne pas avoir à paramétrer chaque copie de QGis qui ira les consulter. Tout cela se réalise maintenant automatiquement à l'ouverture de la couche. En outre, la couleur d'une couche comporte maintenant quatre composantes : RVB + alpha. La transparence peut donc se régler de manière globale, ou bien couleur par couleur, c'est-à-dire par poste.*

Par ailleurs, l'ajout de la bibliothèque de style fait gagner beaucoup de temps, particulièrement quand on cherche à se constituer un nuancier standard que l'on réutilise systématiquement. L'organisation de cette bibliothèque en groupes puis sous-groupes est également très pratique ! » En sus, *QGis 2* permet maintenant d'importer des couches depuis un autre projet. Le seul bémol concerne la légende, qui, pour l'instant, ne répercute pas toutes les possibilités de symbologie, ce qui peut provoquer un décalage entre carte et légende.



Le moteur d'étiquettes prend en compte des caractéristiques dynamiques, par exemple la population, pour régler les caractéristiques de la police d'affichage (illustration Alisé Géomatique).

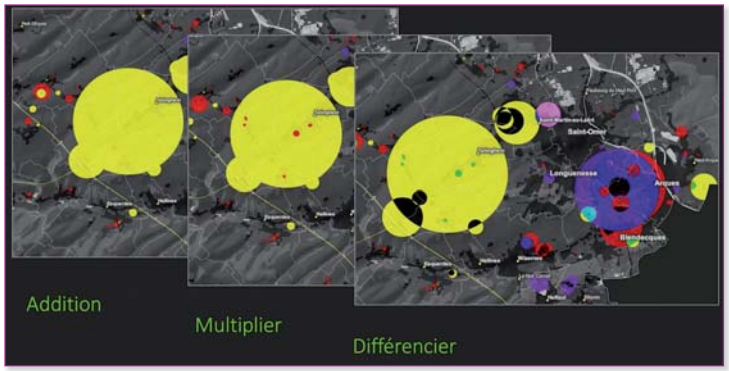
Le moteur d'étiquetage n'a pas non plus été oublié. Il a lui aussi été enrichi de multiples options nouvelles, parmi lesquelles le formatage de texte « *à la volée* » qui offre la possibilité, par exemple, de représenter en capitales des objets dont un des attributs correspond à un critère donné ; comme pour le style, il est également possible de créer des champs dynamiquement, en fonction de la présence ou de l'absence d'une donnée attributaire. Des effets visuels comme le halo, le détourage, etc. améliorent la lisibilité du texte quel que soit l'arrière plan.

Un opérateur de fusion (*blending mode*) peut être également associé à la symbologie. De quoi s'agit-il ? « *D'un moyen élégant de gérer les chevauchements*, poursuit Hélène Durand. *Supposons que vous représentiez la population de chaque ville par un cercle proportionnel. Deux grandes villes proches – par exemple une préfecture et sa voisine – sont susceptibles de se chevaucher. Grâce au choix de la fusion « différence », par exemple, vous obtenez une lentille d'intersection transparente, ce qui améliore grandement la lisibilité ! Il serait également possible de jouer sur la couleur grâce à d'autres options de fusion, mais je ne crois pas que cela soit très pertinent, car on risque la confusion sémiologique.*

Enfin, un autre avantage est la parfaite intégration entre QGis bureautique et QGis Serveur, ce qui permet de publier sur le web des cartes utilisant exactement les mêmes symbologies, et donc d'obtenir un rendu strictement identique sur n'importe quel client léger. »

DB Manager

Pour ceux qui utilisent *QGis* conjointement avec une base *PostGIS* ou *SpatiaLite*, l'ajout du module *DB Manager* sera sans doute le bienvenu. Jusqu'ici, il était très difficile, voire impossible d'agir sur la base, par exemple pour triturer des valeurs, sans avoir recours à un utilitaire extérieur, de type *PGAdmin*. Le module *DB Manager*, construit en *Python* au-dessus de la bibliothèque *Psycopg*, résout élégamment cette entorse à l'ergonomie. Sans aller aussi loin que *PGAdmin III* en termes de possibilités offertes, il permet d'explorer les différentes bases et schémas, de visualiser le contenu des tables, d'éditer celles-ci, d'en changer la structure, voire d'ajouter des contraintes, comme la création de clefs primaires pour les tables qui n'en contiendraient pas.



Les opérations de fusion entre symboles permettent de s'affranchir des difficultés de lisibilité liées aux chevauchements et/ou aux recouvrements (illustration Alisé Géomatique).

Le *DB Manager* propose également un éditeur *SQL* avec colorisation des mots-clefs et autocomplétion. Ce dernier est suffisamment intelligent pour récupérer la structure des schémas et des tables, et connaître ainsi le nom de toutes les combinaisons possibles schéma/table/colonne. Le résultat d'une requête SQL est soit affiché sous forme de tableau, soit récupérable directement sous forme de couche *QGis*. Toutefois, cette dernière option ne crée pas de table, si bien que le résultat de la requête est perdu si la couche est supprimée. Il faut donc recourir à la syntaxe classique *CREATE TABLE nom AS (SELECT…)* si l'on veut enregistrer le résultat définitivement dans la base - dans ce cas, la requête ne renvoyant aucun résultat, il faut donc explicitement charger la couche par l'intermédiaire du menu/icône pour l'afficher.

QGis, plate-forme de développement

La conception modulaire de *QGis* ouvre la voie à plusieurs types de programmation. Il y a tout d'abord l'extension traditionnelle, écrite au-dessus de l'API *Python*, et qui publie généralement un ou plusieurs services utilisables dans l'environnement *QGis*. Dans ce cas, le code *Python* se sert de l'API pour accéder aux données et afficher les résultats. Cette possibilité convient pour des applicatifs courts et réutilisables.

Pour des applications plus étoffées, plus orientées métier et personnalisant l'interface graphique de *QGis* pour la rendre plus appropriée à l'utilisation prévue, on peut se tourner vers l'utilisation de l'API comme véritable plate-forme de programmation. C'est ce qu'a entrepris Sylvain Pierre, ingénieur géographe à la direction de l'agriculture, de l'espace rural et de l'environnement du Conseil général du Bas-Rhin.

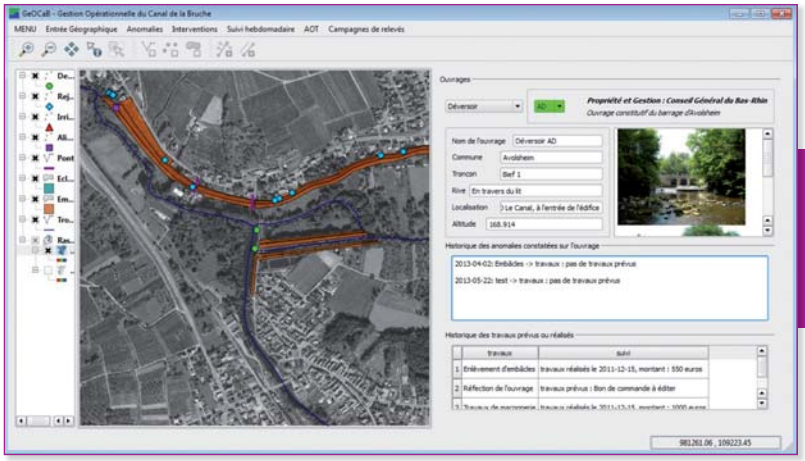
« *L'origine de notre projet,* explique-t-il*, remonte au transfert de propriété du Canal de la Bruche de l'État vers le département. Ce fut, pour ainsi dire, la goutte d'eau, car la plupart des agents manifestaient déjà une forte lassitude envers les solutions SIG existantes. Nous avons donc décidé de partir sur des bases nouvelles. Pour cela, nous avons évalué la performance et la faisabilité d'un certain nombre de configurations fondées sur de l'existant ou du logiciel libre. Le couple QGis/SpatiaLite est arrivé en tête, nous avons donc décidé de franchir le Rubicon et de réaliser un développement en interne.* »

Une absence de contraintes

À l'époque du démarrage du développement, il s'agit encore de la version 1.8 de *QGis* ; le choix de *SpatiaLite* s'explique par la faible empreinte de la base de données, couplée à une installation particulièrement simple qui s'effectue en local sur le poste de développement, et ne requiert donc pas d'intervention particulière de la DSI. « *Le département utilise une base* Oracle*, à laquelle QGis 1.8 ne peut pas se connecter. Nous n'avons découvert qu'après coup que nous disposions d'une base* PostGIS *librement accessible. Nous avons donc basculé de* SpatiaLite *vers* PostGIS *en cours de projet*, indique Sylvain Pierre.

À partir du moment où nous avons décidé d'entreprendre le développement de zéro, nous avions le choix entre écrire un simple plug-in *ou une application métier totalement dédiée. Nous avons opté pour cette deuxième possibilité, considérant que notre application était trop spécialisée pour mériter une diffusion large : notre modèle de données reflétait la particularité du contexte, orienté autour de la saisie des incidents (dégâts à l'infrastructure, recensement des fuites, des arbres tombés dans le canal…), la gestion des interventions (bons de commande, programmation des travaux…), la gestion du domaine public (autorisations d'occupation temporaire) plus l'aspect environnemental comme le recensement des milieux naturels, l'inventaire des essences de la ripisylve, etc.*

La partie géographique possède un lien très fort avec les données alphanumériques, raison pour



L'interface de l'application *Python* spécialisée pour la gestion du canal de la Bruche. La partie *QGis* se limite à la fenêtre de visualisation, le gestionnaire de couches, par exemple, a été entièrement reprogrammé (il ne ressemble d'ailleurs que superficiellement à celui de *QGis*).

laquelle nous sommes partis d'emblée sur une solution bureautique plutôt que d'opter pour un client léger de type web *conçu à l'aide d'une des nombreuses et excellentes bibliothèques* Javascript *existantes.* »

Écrire une application métier au-dessus de *QGis*, cela signifie utiliser l'API du logiciel comme une simple bibliothèque *Python*, et donc devoir écrire sa propre interface graphique en utilisant, par exemple, la bibliothèque *PyQt*, qui permet d'appeler depuis *Python* les services offerts par *Qt*, lequel est conçu en C++. « *L'effort initial consacré à l'apprentissage de* Python *n'est pas négligeable*, admet Sylvain Pierre, *mais celui-ci est facilité par le grand nombre de sites* Internet *disponibles proposant des méthodes d'apprentissage et/ou des trucs et astuces.* Python *possède, à mes yeux, des avantages majeurs : sa syntaxe minimaliste évite de s'encombrer l'esprit avec des « tournures » particulières, et sa structure en fait un langage attrayant, sans compter le fait qu'étant interprété, la mise au point se réalise très facilement. La maîtrise de* Qt *est en revanche un peu plus délicate, particulièrement les notions de signaux/fiches qui constituent l'essence de l'interaction. Mais là aussi, avec un peu de pratique, on prend vite le pli. Le dernier élément à connaître, c'est bien sur l'API* Python *de* QGis. »

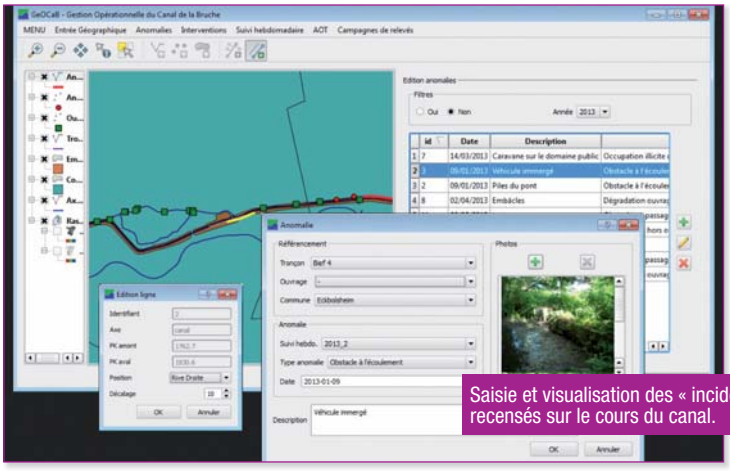
Le développement prend environ quatre mois pour une équipe de deux personnes (six à sept mois/homme). Le produit une fois achevé, quelles sont les leçons à tirer du projet ? « *Déjà, nous avons dû nous former à* Python *et à l'approche objet, une technique de programmation à laquelle nous n'étions absolument pas formés. Après coup, je dirais que celle-ci est très intuitive, même si, là aussi, il faut consentir un effort initial certain. Ensuite, l'utilisation conjointe des divers composants Open Source mis en œuvre permet de mieux cerner l'architecture d'un SIG, c'est-à-dire ce qui est dévolu au code, à la base de données ou encore à la bibliothèque de représentation. Le sentiment général se traduit par une absence de contraintes, l'impression que l'on peut, plus ou moins, réaliser tout ce que l'on souhaite, sans que les choix effectués soient soumis à ceux imposés par tel ou tel éditeur. Développer sa solution revient à se (ré)approprier la technologie.* »

Vers la version 2.0

Parmi les améliorations prévues figure, bien entendu, le passage à la version 2.0. Ce dernier ne va pas tout à fait de soi - quoique l'effort ne soit pas très important - puisque l'API a changé.

« *Mais c'est un effort que nous consentons volontiers*, explique Sylvain Pierre, *car la possibilité de se connecter à une base* Oracle *va nous permettre d'enrichir la fenêtre cartographique avec des données issues du serveur départemental.* » Avec, en ligne de mire, le développement d'une véritable application de gestion des rivières en général, capable de gérer et d'enregistrer des données depuis tout type d'environnement. « *Le développement à façon d'applications métier constitue peut-être un moyen de contourner la marginalité du SIG en termes d'utilisateurs spécialisés. Il faut bien reconnaître que, dès que l'on sort de la sphère du grand public avec les outils que l'on connaît, comme* Google Earth, *par exemple, le nombre de non-géomaticiens utilisant des SIG chute considérablement. Je pense que ces programmes, où le SIG apparaît comme sous-jacent, sont un bon moyen de familiariser certains experts avec les usages et les avantages de la géomatique.*

C'est pourquoi, dès lors que nous disposerons d'un code acceptable - propre et documenté - j'envisage de le publier. Mais à quelle échéance… je préfère ne pas me prononcer ! », conclut Sylvain Pierre. I



Saisie et visualisation des « incidents » recensés sur le cours du canal.