Intégration d'infos-trafic temps réel dans un moteur de calcul d'itinéraires

Projet des élèves ingénieur de l'ENSG

A. Pauthonnier ■ *ENSG – alexandre.pauthonnier@ensg.eu*

Les élèves ingénieurs de 3e année de l'*École Nationale des Sciences Géographiques* (ENSG), spécialisés en architecture de solutions SIG (filière ASIG), ont consacré leur dernier mois dans les murs de l'ENSG à un projet de fin d'études proposé par *PTV Loxane*, éditeur de solutions logicielles, *leader* dans le secteur de la planification et de l'optimisation de déplacements. Ce projet industriel s'inscrit dans le cadre du trophée étudiant de l'intégration, concours organisé tous les ans par *PTV Loxane* et pour lequel les résultats seront connus à l'automne. La société fournit des services sous forme de composants logiciels à des intégrateurs qui développent des applications métiers (logistique, transport). L'offre de services (*route planner* notamment) s'appuie sur deux types de technologies dont la technologie *ActiveX* ; celle-ci intègre un système de couches de restrictions qui – et c'est l'idée du projet – pourrait être utilisé à terme, pour porter de l'information trafic temps réel, et ce quelle qu'en soit la source ; les intégrateurs sont évidemment preneurs, la valeur ajoutée de ces informations n'étant pas à démontrer.

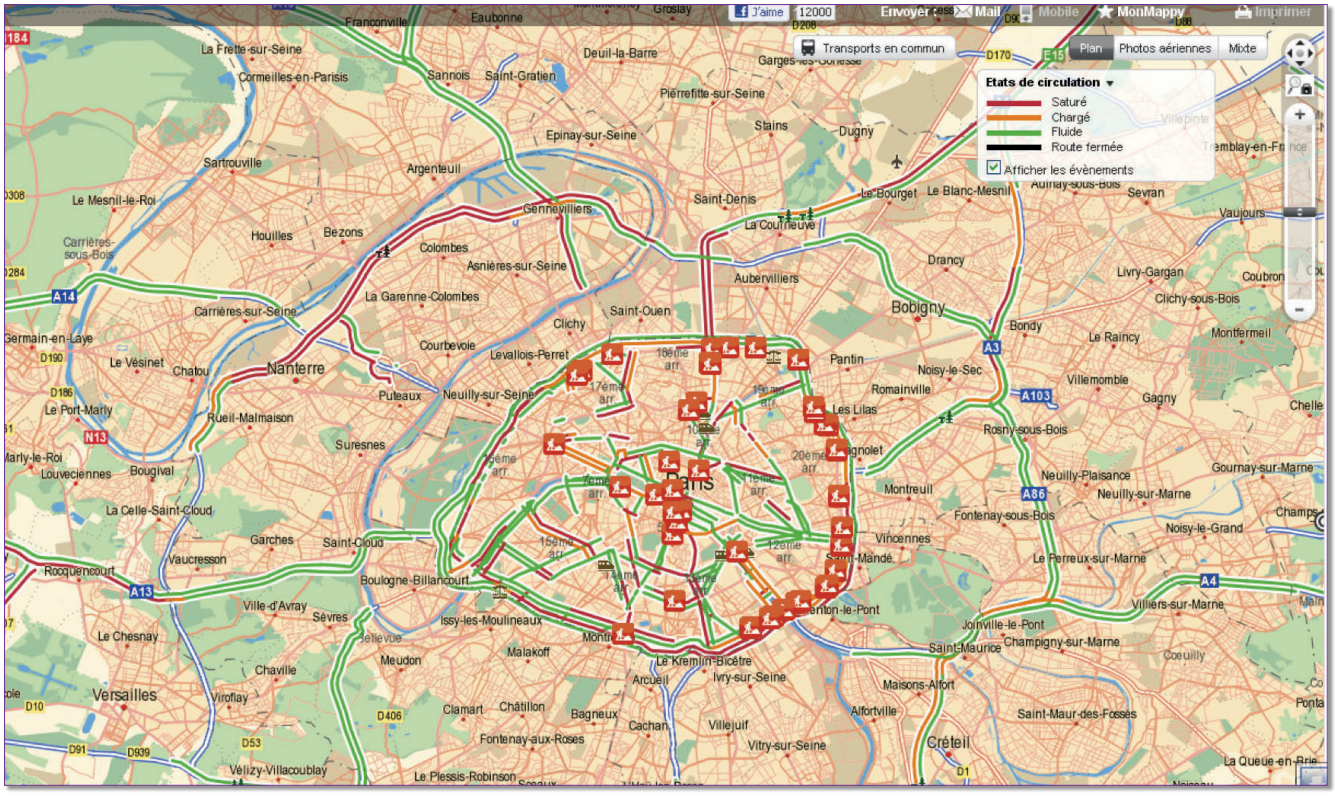


Figure 1 – État du trafic en région parisienne le 23/06/2011 à 10:00 sur le portail web de Mappy – Affichage des évènements.

Ces données, dont la collecte et la diffusion ont commencé il y a une quinzaine d'années sont aujourd'hui largement utilisées par les systèmes de navigation embarquée qui les captent en modulation de fréquence mais aussi depuis peu par des clients *web*. Les quatre étudiants, *Maïmouna Deme*, *Claire Lagesse*, *Francis Bock* et *Ugo Fillastre* ont donc développé un prototype

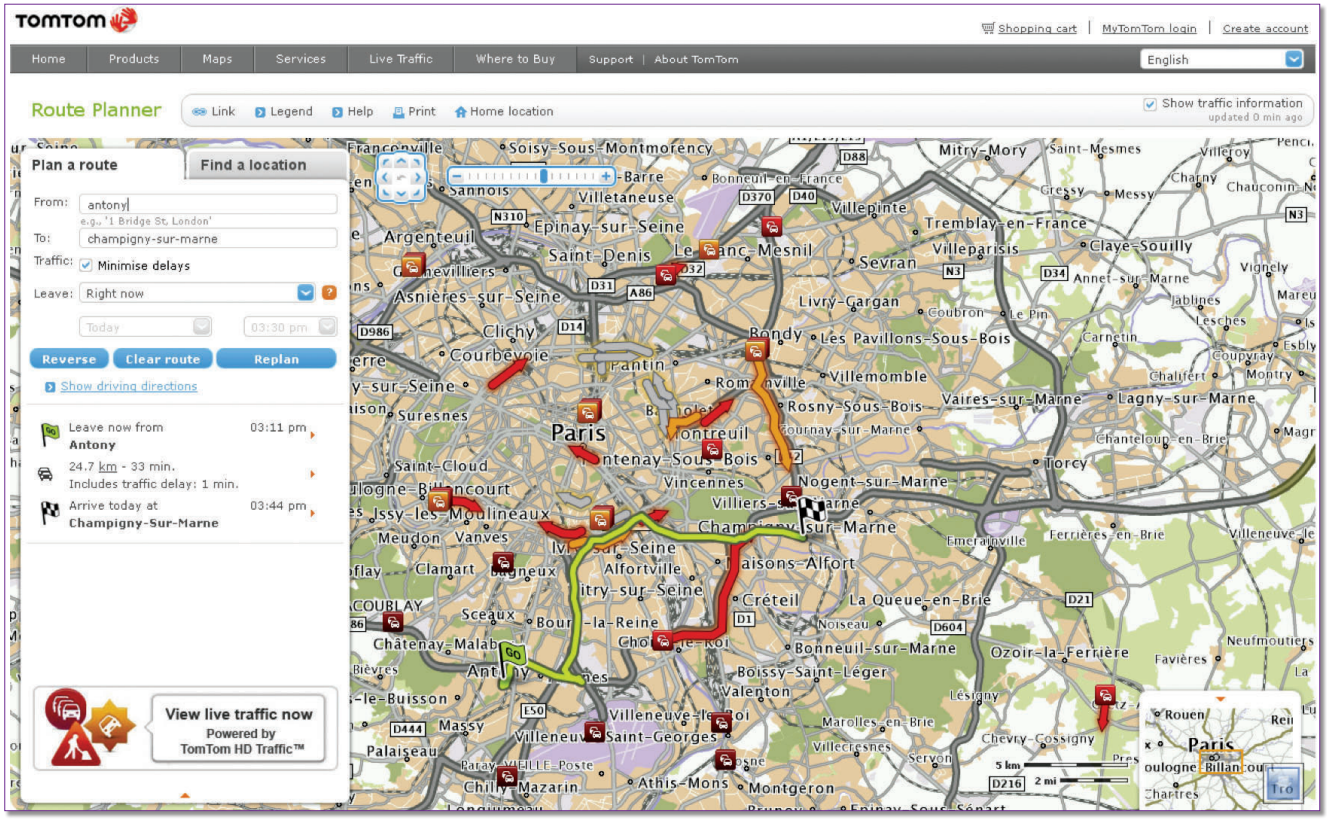


Figure 2 – Calcul sur le portail web de TomTom de l'itinéraire reliant Antony (92) à Champigny-sur-Marne (94). En raison d'un accident au niveau de Choisy-le-Roi, un déroutage est proposé.

permettant de valider le principe de l'utilisation des restrictions pour intégrer ces données dans les composants.

L'information trafic temps réel et la géomatique

Les systèmes d'aide à la navigation ont connu une croissance très importante durant la dernière décennie. Qu'ils soient intégrés au tableau de bord du véhicule, qu'ils se présentent sous forme de boîtiers autonomes (*Tom-Tom*) ou bien, pour les plus récents, qu'ils fonctionnent sur *smartphones*, ils ont été adoptés par une grande partie des automobilistes. À ces outils embarqués, il faut ajouter les outils destinés aux internautes, les plus populaires étant les portails *web* de *Bison futé*, *Mappy*, *Mediamobile* (*http://www.v-trafic.com*), *ViaMichelin*, *Tom-Tom* ou encore celui de *Google Maps* qui tous proposent (depuis plus de dix ans pour certains) des outils simples d'emploi pour préparer efficacement un déplacement : calcul de l'itinéraire le plus rapide, le moins cher, le plus touristique (Figure 1) Soulignons que ces différents outils ont largement contribué à rendre populaire le mariage de la cartographie, de l'informatique et du GPS, bref à rendre populaire la géomatique.

Ces systèmes reposent sur des bases de données routières régulièrement actualisées mais aussi sur des informations temps réel caractérisant l'état du trafic et signalant les différentes perturbations affectant la circulation (travaux, incidents, congestion, phénomènes météorologiques). Bien entendu, ces infos-trafic améliorent le service rendu par les *route planner* classiques : désormais l'outil ne calcule plus un optimum théorique mais bien un optimum à l'instant *t* ; sur la foi de ces informations temps réel, le système de navigation devient capable de recalculer à tout moment le temps de parcours pour aller d'un point *A* à un point *B*, voire de dérouter le véhicule sur un *itinéraire bis* si l'optimum proposé initialement ne l'est plus (Figure 2).

Les informations temps réel exploitées dans les systèmes de navigation sont issues d'une chaîne de collecte et de traitement multi-sources ; celles-ci s'appuient sur une organisation relativement complexe qui gravite autour des pouvoirs publics. Depuis une trentaine d'années, ces derniers

mesurent l'impact des problèmes de transport routier sur l'économie, l'environnement, la santé, la sécurité ; l'organisation des transports s'est imposée comme un enjeu politique important. Face à une augmentation du trafic, les pouvoirs publics (l'État, l'Union européenne), confrontés à un problème *a priori* sans solution, ont vu, à la fin des années 90, dans les systèmes de navigation une aubaine : ils ont alors investi largement pour favoriser leur développement. Tout d'abord en finançant l'équipement des axes autoroutiers en systèmes de collecte de l'information trafic (boucles électromagnétiques intégrées au goudron de la chaussée, caméras), mais aussi en mettant en place des structures pour recueillir et diffuser cette information : le CNIR de Rosny-sous-Bois s'appuyant sur les CRIR (Centres régionaux d'information routière), *Bison futé*, la mascotte des départs en congés, qui depuis 1976 dessine en vert, orange, rouge ou noir les routes de France. Enfin, en attribuant des missions *« infos-trafic »* aux services de police, de gendarmerie mais aussi aux collectivités locales.

Aujourd'hui, tous les systèmes de navigation exploitent ces données publiques, mais ils bénéficient en outre d'autres d'informations, fournies par des sociétés privées : celles-ci gèrent notamment des flottes de véhicules traceurs équipés de GPS (éventuellement intégrés à des téléphones mobiles). La multiplication des sources a ainsi permis de densifier largement les axes couverts (de 20 000 km au début des années 2000 à plus de 300 000 km aujourd'hui).

Cela étant, l'exploitation simultanée de toutes ces informations par les systèmes de navigation embarquée ne va pas de soi : de nombreux pré-traitements, incluant des phases de synchronisation spatiale et temporelle entre les sources, de filtrage, d'enrichissement et d'encodage selon un standard partagé, sont nécessaires, le tout dans un délai compatible avec une diffusion en quasi temps réel, ce qui est loin d'être simple ! Cette tâche est dévolue à des diffuseurs (*Mediamobile*, *ViaMichelin* par exemple) qui alimentent *in fine* les systèmes de navigation en transmettant l'information par radiodiffusion FM ou par les réseaux de téléphonie mobile. Jusqu'à peu, c'étaient les seuls moyens de diffusion ; la généralisation de l'accès *web* et des technologies associées (services *web*) a un peu changé la donne : il est intéressant de noter qu'aujourd'hui, les diffuseurs sont également devenus des fournisseurs de données. À partir des informations collectées, ils ont élaboré des modèles statistiques permettant de produire des profils de fréquentation horaire par axe routier et ce pour chaque jour et mois de l'année. L'information trafic, telle qu'on la consomme aujourd'hui (chez *Google*, par exemple), mélange ainsi information temps réel et information statistique.

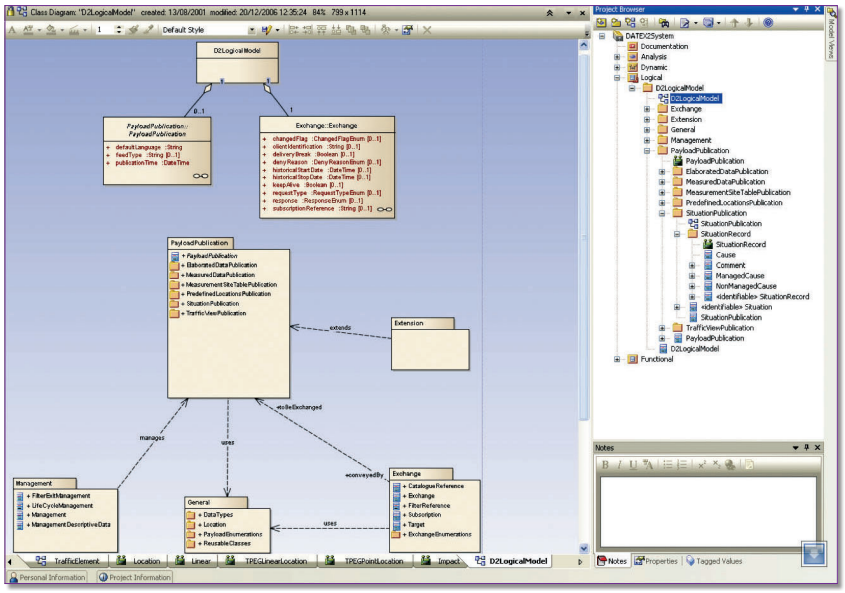


Figure 3 – Vue d'ensemble du diagramme de classes Datex.

Un standard européen

L'essor des services de navigation repose en grande partie sur un travail de standardisation mené au niveau européen : le standard *Datex*, publié à la fin des années 90, a fourni un cadre commun aux fournisseurs de données et de services pour manipuler numériquement les données d'information routière. Une version modernisée, *Datex2* est actuellement en phase de *publication* : elle doit corriger certains défauts de la version 1 qui s'est avérée finalement peu efficace sur les questions d'interopérabilité. Or, cet enjeu est fondamental, les données d'infos-trafic ayant vocation à être exploitées non seulement par les opérateurs de services (les centres de gestion du trafic, les centres d'information routière) mais aussi bien sûr par les diffuseurs et par tous les usagers de la route (professionnels ou grand public). *Datex2* s'appuie sur trois piliers fondamentaux :

- La notion de modèle conceptuel de données (formalisme UML) indépendant de toute plate-forme d'implémentation ;
- Une séparation rigoureuse entre les données décrivant le trafic et celles utiles à la mise en œuvre des échanges ;
- L'utilisation des technologies *web* pour l'échange des données (XML, HTTP et services *web*).

Ces nouvelles spécifications n'ont pas laissé nos étudiants ENSG insensibles : plus que tout, ils savent qu'il n'y a pas de problème géographique bien posé, et *a fortiori* de solution à ce problème, sans modèle conceptuel de données ! Les technologies *web* sont par ailleurs au cœur de leur formation ; ils ont les connaissances nécessaires pour comprendre et mettre en œuvre des solutions SIG qui les exploitent. Le modèle UML est librement téléchargeable sur le site *web* *Datex* (http://www.DATEX2.eu). Il est composé de deux diagrammes : les diagrammes de classes (ou diagrammes logiques) qui décrivent les concepts et les diagrammes de comportements (dynamiques et fonctionnels) qui décrivent les mécanismes d'utilisation des concepts (essentiellement les échanges de données d'infos-trafic entre le fournisseur et le client). Les classes sont regroupées en quatre packages (Figure 3).

Toute la richesse de la syntaxe UML n'est pas utilisée, mais on retrouve les éléments fondamentaux du langage : notions de classe, d'attribut, de lien valué entre classes, de lien d'héritage, d'agrégation ou de composition. *Datex2* utilise le concept générique de *publication* pour diffuser les données d'infos-trafic ; ce concept est décliné de quatre manières.

Les données peuvent par exemple être publiées sous forme d'objets *situation* (on parle alors d'une *situation publication*) ; un objet *situation* regroupe plusieurs éléments caractérisant les conditions de circulation à un endroit donné du réseau ; chaque élément relatif à une *situation* est un objet *situation record*. C'est le niveau du modèle UML qui permet d'identifier par un objet géoréférencé (qu'on appellera évènement par la suite) un accident, un incident, des travaux, un bouchon… (Figures 4 et 5).

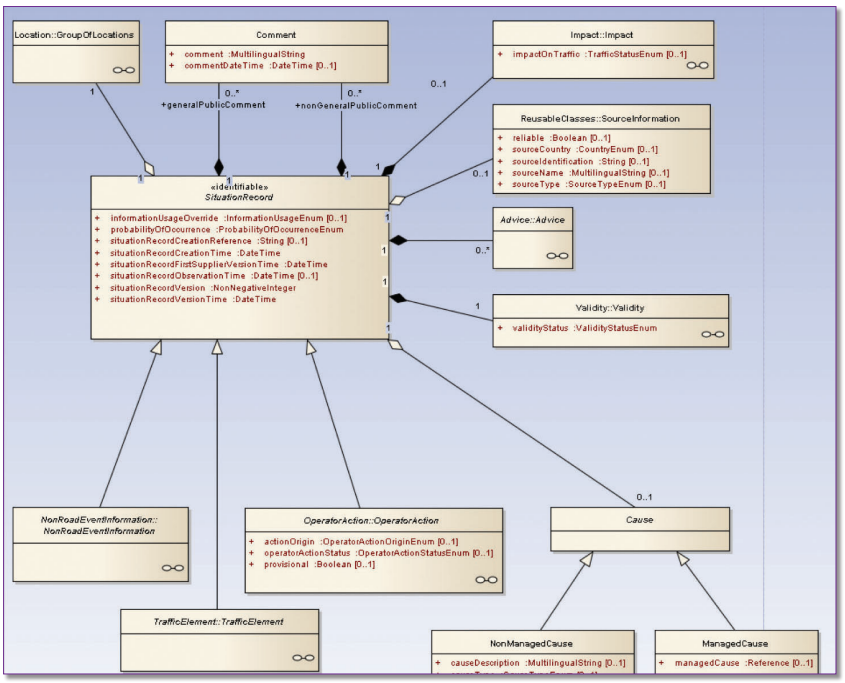


Figure 4 – Extrait du diagramme de classes Datex montrant le concept d'évènement routier (situation record).

Tous ces évènements sont géolocalisés. *Datex* propose plusieurs méthodes dont le repérage par segment de route : dans ce mode de référencement, chaque évènement est associé à un nœud initial (*objet from*) et à un nœud final (*objet to*), chacun étant localisé en coordonnées géographiques dans le système ETRS89 ; il porte le nom du tronçon de route concerné. À ces éléments peuvent être associés des informations sémantiques qualifiant la perturbation : nature, cause, données d'impact (classe *impact*) et notamment les notions de *capacité restante* (33 % s'il n'y a plus qu'une voie opérationnelle sur trois) et de délai (temps qu'il faut ajouter au temps de parcours habituel). Dans ce style de *publication*, chaque objet *situation* est *« historisé »* : à chaque mise à jour, si la *situation* est toujours d'actualité, une nouvelle version est créée.

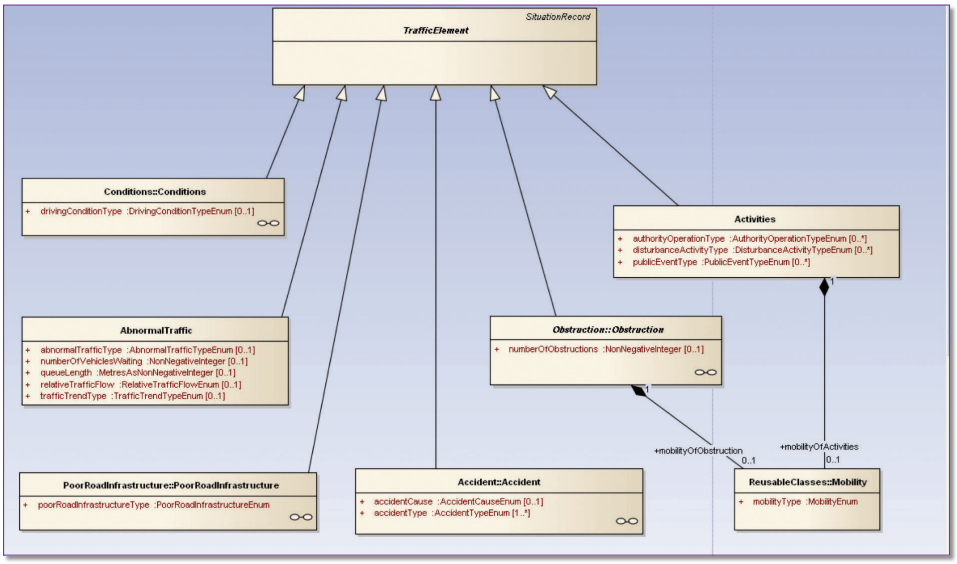


Figure 5 – Extrait du diagramme de classes Datex montrant les différents types d'évènements modélisés.

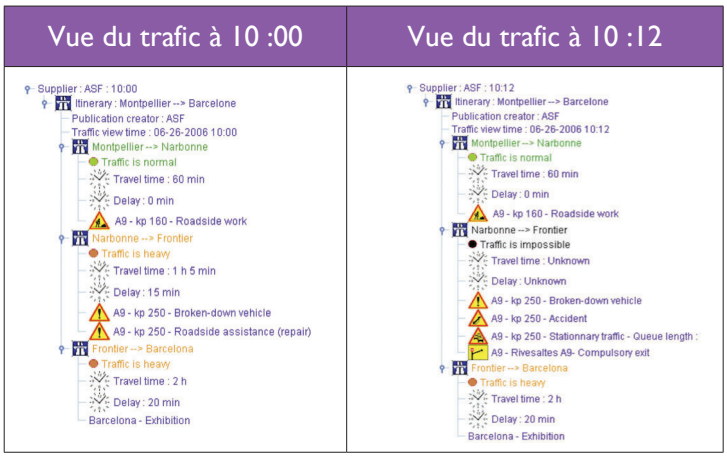


Figure 6 – Publication par itinéraire routier d'infos-trafic selon le standard Datex.

Datex2 permet également de publier les données sous forme d'une *traffic view publication*, laquelle propose des instantanés des conditions de trafic par itinéraire routier. C'est une autre approche, qui s'appuie notamment sur la notion de *sections* d'itinéraires et qui permet d'accéder à une information plus directement exploitable par l'utilisateur final (Figure 6).

Les aspects fonctionnels et dynamiques sont regroupés dans des diagrammes de classes intégrant la notion de méthode ainsi que dans des diagrammes de séquence illustrant les différents mécanismes d'échange. *Datex2* identifie deux types d'acteurs (fournisseur et client) et propose trois types de services : *subscribe*, *push* et *pull* (Figure 7). *Push* permet au fournisseur de diffuser ses données : il peut le faire à chaque fois qu'il y a une modification dans les données (*PushOnOccurrence*) ou bien régulièrement selon une fréquence prédéfinie (*PushPeriodic*). *Pull* permet au client de télécharger les données. *Subscribe* permet à l'un comme à l'autre de définir le type de données souhaitées.

Pour mettre en œuvre ces services, deux types d'implémentation sont envisageables : diffusion des ressources XML par HTTP (requêtes GET) ou encore via des *web* services SOAP sur HTTP .Le site *Bison-Futé* permet actuellement d'accéder à titre expérimental à des flux XML *Datex2*. Un des flux proposés utilise le format de *publication* par évènements : dans la mesure où il couvre le territoire national c'est lui qui a été retenu par les étudiants pour alimenter le prototype.

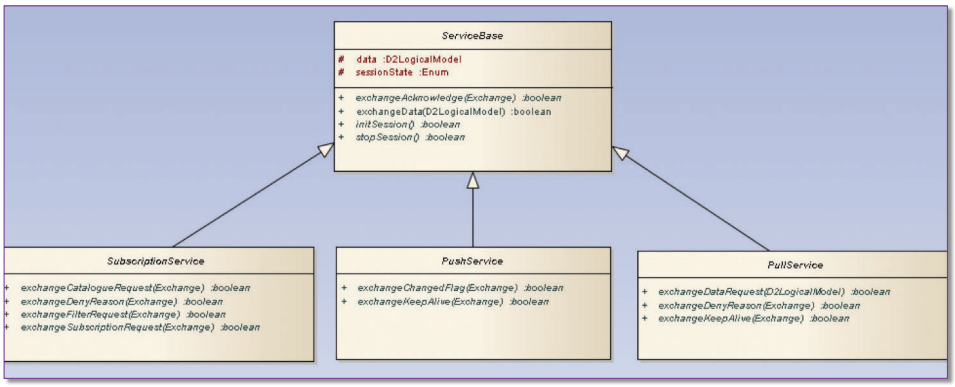
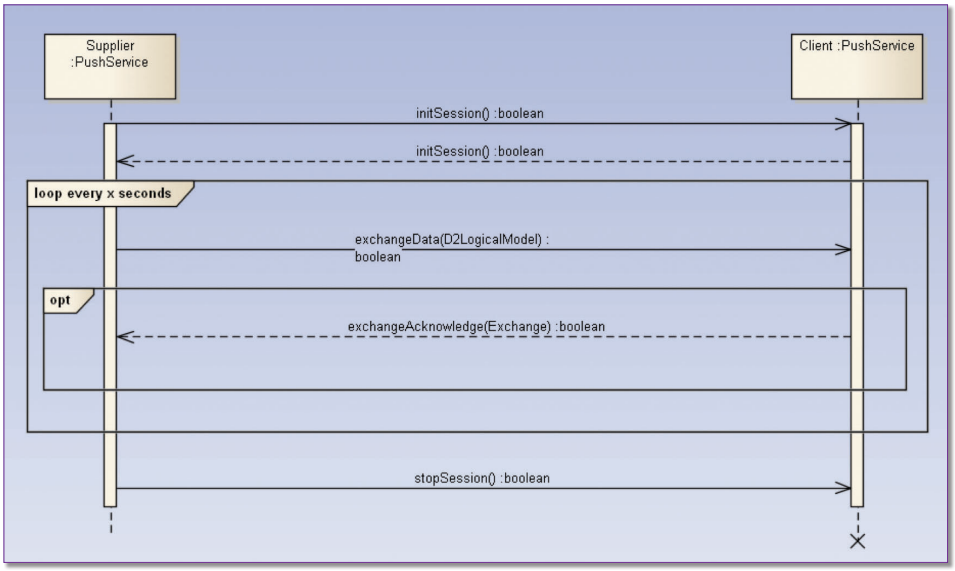


Figure 7 – Diagramme de classes modélisant les échanges de données Datex entre acteurs – Diagramme de séquence du service PushPeriodic.



Les composants Loxane X

Ainsi que nous l'avons vu en introduction, *PTV Loxane* fournit des services de cartographie et de calcul d'itinéraires à des intégrateurs qui développent des applications métier, utilisées en situation opérationnelle (suivi d'une flotte de véhicules et routage dynamique) ou bien en anticipation (optimisation de tournées). Actuellement, la société allemande dispose d'une gamme de produits s'appuyant sur deux types de composants logiciels : les composants *Loxane X* qui sont en fait des contrôles *ActiveX* (composants logiciels réutilisables et intégrables par développement dans des applications tierces)

et les *PTV xServers* qui sont des *web* services SOAP. Chacune des deux technologies, adaptée à des contextes d'utilisation différents (applications bureautiques ou client intranet pour *Loxane X*, client *web* pour *PTV xServers*) offre des services de cartographie, de géocodage et de calcul d'itinéraires conçus pour s'adapter aux multiples contextes d'utilisation professionnelle (prise en compte des contraintes de circulation des poids-lourds par exemple). Les composants *ActiveX* sont utilisés à la fois par des bureaux d'études (optimiser des tournées de ramassage ou d'approvisionnement) et par des services opérationnels : le SAMU, par exemple, les exploite en relation avec ses systèmes embarqués pour aider à l'identification du véhicule de secours le plus proche ; un seul bémol : *a contrario* des *PTV xServers*, ces outils ne bénéficient pas encore de l'information trafic temps réel.

Comme tout contrôle *ActiveX*, *Loxane X* est une sorte de bibliothèque compilée autonome dotée d'une interface graphique, que l'on peut manipuler à l'aide d'une API : on en compte quatre ; elles sont accessibles en environnement C++, VB, *Delphi*... Dans le cadre du projet, les étudiants ont développé avec *Visual Studio C++*. Les outils *Loxane X* sont conçus pour interagir avec des bases de données propriétaires de l'éditeur archivant au format vecteur (points, linéaires et surfaciques) des données de fond de carte ainsi que des graphes routiers. Ces données couvrent la France mais aussi la plupart des pays européens.

• *Loxane X MapDisplay* offre des services de cartographie : il permet d'afficher dans une fenêtre cartographique les données de fond de plan (appelons la *« BD rendering »*) ainsi que des données utilisateur en superposition. L'interface permet le dialogue homme machine (clic, déplacement, sélection…) ;

• *Loxane X DataAccess* permet de manipuler les bases de données propriétaires à des fins d'analyse (sur critères sémantiques et/ou spatiaux) ou encore d'édition (saisie, mise à jour, suppression). Il est doté d'une interface graphique de type tableur qui

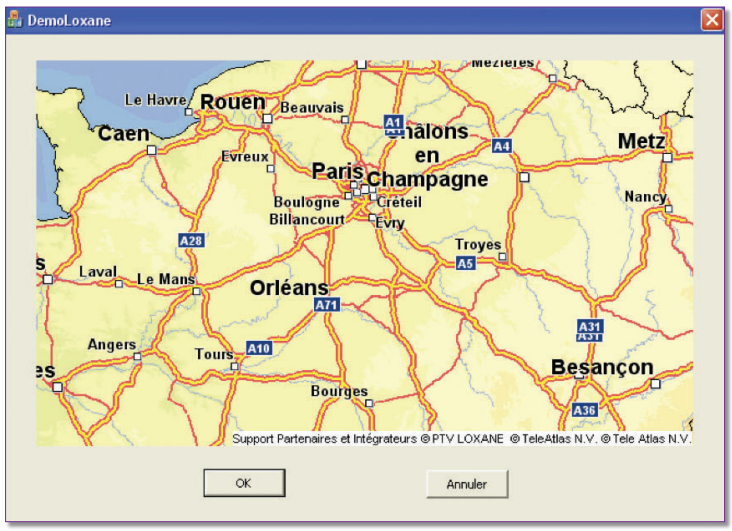


Figure 8 – Intégration de Loxane X MapDisplay dans une application MFC.

permet de visualiser la sémantique des objets. *DataAccess* dialogue avec *MapDisplay* pour afficher dans la fenêtre carte les objets traités (sélection) ;

• *Loxane X RouteFinder* offre des services de calcul d'itinéraires et d'isochrones. Il affiche l'itinéraire calculé sous forme d'un tableur donnant soit les caractéristiques globales du parcours (kilométrage, temps de parcours, coût, consommation de carburant) soit les éléments détaillés (feuille de route montrant les étapes). Il interagit avec *MapDisplay* afin de faire afficher sur le fond de carte l'itinéraire et ses étapes. Pour réaliser les calculs, *RouteFinder* manipule un graphe (topologie de type *arc-nœud*) archivé de manière optimisée dans une base de données propriétaire (appelons la *« BD routing »*). Le graphe est en fait constitué de plusieurs sous-graphes interconnectés (un graphe par pays et catégorie de route) ; à chaque arc est associé une sémantique décrivant les conditions de circulation : un booléen de validité (on peut circuler ou pas) et une vitesse dérivée des caractéristiques de la route.

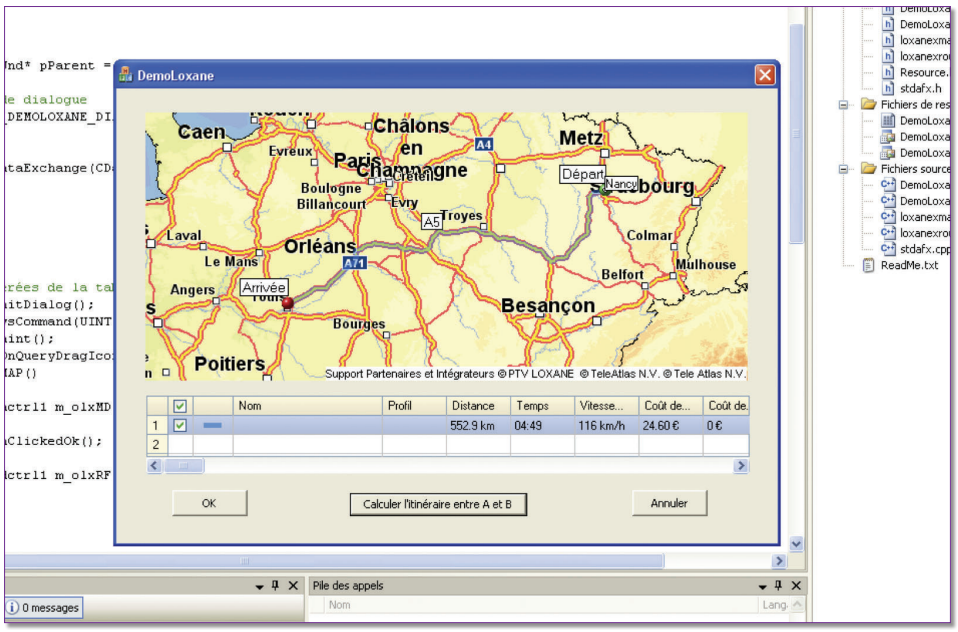


Figure 9 – Intégration et utilisation Loxane X MapDisplay et RouteFinder dans une application MFC.

Un mécanisme dit *« de couche de restriction »* permet de modifier ces valeurs : il consiste à copier dans une base de données de *restriction* les éléments du graphe dont il faut modifier la sémantique ; une fois ces restrictions créées et activées, *RouteFinder* en tient compte lorsqu'il parcourt le graphe. C'est ce qui permet au composant à partir d'une même *BD routing* d'implémenter des conditions de navigation propres à chaque type d'utilisateur (transports poids-lourds, intervention d'urgence…) ;

• Mentionnons *Loxane X AdressFinder*, le quatrième, qui fournit des services de géocodage mais qui n'a pas été utilisé dans ce projet.

Pour prendre en main les composants, il suffit, après les avoir installés sur le système d'exploitation hôte, de créer une application graphique de type *Windows MFC* puis d'ajouter à l'interface de l'application le ou les contrôles *ActiveX* souhaité(s). Les assistants permettent ensuite de déclarer chaque contrôle comme une variable membre de l'application et de créer à la volée pour chacun une classe d'encapsulation C++ qui va permettre d'utiliser le composant comme s'il s'agissait d'un objet C++ natif. Reste alors à initialiser chaque *Loxane X* dans le constructeur de l'application ; pour *MapDisplay*, trois lignes de code suffisent : lecture d'un fichier de configuration XML (décrivant les différentes couches de données à afficher, leur représentation graphique), centrage de la carte et enfin affichage de celle-ci. À l'exécution, on obtient une interface contenant une fenêtre cartographique dans laquelle on peut se déplacer à volonté ; en fonction de l'échelle d'affichage, *MapDisplay* choisit les couches à afficher ainsi qu'une symbologie appropriée (Figure 8a/8b).

Pour initialiser les autres composants (*RouteFinder* notamment), le principe est identique : leur utilisation nécessite de bien connaître les API mises à disposition. La figure 9 montre l'interface d'une application MFC intégrant à la fois *MapDisplay* et *RouteFinder* : elle permet d'utiliser la souris pour indiquer un lieu de départ et un lieu d'arrivée, de calculer l'itinéraire le plus rapide les reliant et d'afficher à la fois sous forme d'un tableau et d'un objet graphique l'itinéraire calculé. Une trentaine de lignes de code suffisent, la partie délicate pour le non-initié à la technologie *ActiveX* étant sans doute l'identification des mécanismes d'interception par l'application MFC des évènements portant sur les contrôles.

Analyse et proposition d'une solution

Le mécanisme des restrictions permet de modifier à volonté les caractéristiques des graphes sans modifier les graphes eux-mêmes. L'idée qui s'est imposée a donc consisté à proposer un moteur permettant de générer, selon une fréquence prédéfinie, une base de données de *restrictions* archivant en temps réel tous les éléments du graphe routier affectés par des évènements perturbant la circulation routière. Le standard *Datex* et, notamment, le format de publication *Situation*, conviennent tout à fait, dans la mesure où les évènements recensés possèdent toutes les informations requises pour créer ces objets *restriction* : géoréférencement, numéro de route, nature de la *restriction*, impact sur le trafic. L'analyse du besoin a ensuite permis d'identifier, parmi les données gérées par le modèle, les informations nécessaires et suffisantes. Ont été retenus pour chaque évènement :

- son identifiant ;
- son type ;
- sa date de début ;
- sa date de fin ;
- le niveau d'impact (exprimé en pour-cent de capacité restante) ;
- le nom de la route affectée ;
- la latitude de début ;
- la longitude de début ;
- la latitude de fin ;
- la longitude de fin.

Ces informations ne constituent qu'une petite partie du modèle, mais ce sont celles qui sont systématiquement remplies dans les flux XML de tests à la disposition du projet. On peut penser que

les futurs services de diffusion, qui s'appuieront sur le standard, les proposeront systématiquement. En ne retenant que ces informations, on s'autorise donc à alimenter les composants *Loxane X* avec plusieurs sources d'infos-trafic. Afin de valider cette idée, le projet a d'ailleurs décidé de créer son propre format interne d'archivage des évènements temps réel et d'alimenter son moteur de calcul des restrictions avec celui-ci (Figure 10).

La création de la base de données de restrictions soulève toutefois un problème bien connu des géomaticiens : celui de l'appariement de données issues de sources différentes. En effet, il s'agit ici de corréler les évènements *Bison-Futé* (géolocalisés en coordonnées géographiques dans le système ETRS89) avec la base de données routières de l'éditeur (référencée dans un système de coordonnées propriétaire). Les différences de système ne sont pas un problème, puisqu'il existe une formule de passage de l'un à l'autre ; en revanche, le vrai obstacle réside dans les différences de spécifications (sémantique, ontologie) entre les deux bases de données.

Certes, toutes deux décrivent le même réseau routier national, mais chacune le décrit à sa manière ; cela signifie très concrètement qu'une même route sera représentée par une géométrie différente. Il s'ensuit qu'un évènement *Bison-Futé* concernant un tronçon de route n'intersectera pas nécessairement le tronçon homologue dans la base de données *Loxane*. Il faut donc rechercher des candidats à l'appariement dans des voisinages, mais aussi exploiter les autres informations à disposition, à savoir le nom de la route et l'orientation du segment de *restriction*. En procédant ainsi, on peut espérer obtenir un bon taux d'appariement, mais vraisemblablement pas 100 %.

Pour illustrer ce souci, prenons l'exemple de travaux réalisés sur l'autoroute A5 sur la commune d'Orges (Haute-Marne). *Bison-Futé* délimite précisément la *restriction* par les points de coordonnées géographiques (5,070 14 ; 48,037 79) et (4,952 92 ; 48,055 91) et indique qu'une des deux voies est inutilisable. De la lecture des coordonnées, on déduit que le problème concerne la chaussée orientée dans le sens Langres-Troyes. Si on visualise les coordonnées de début et fin sur un fond de carte *Google* (Figure 11), on constate sans surprise que le plaquage brut des extrémités du segment *Bison-Futé* n'est pas satisfaisant : un des deux points se trouve sur la chaussée Troyes-Langres et l'autre sur la chaussée Langres-Troyes. Tout ça n'étonnera pas le cartographe averti, qui sait pertinemment qu'une base de données cartographiques (c'est le cas pour le cas pour *Google*, c'est également le cas pour *Loxane*) donne une image généralisée du terrain.

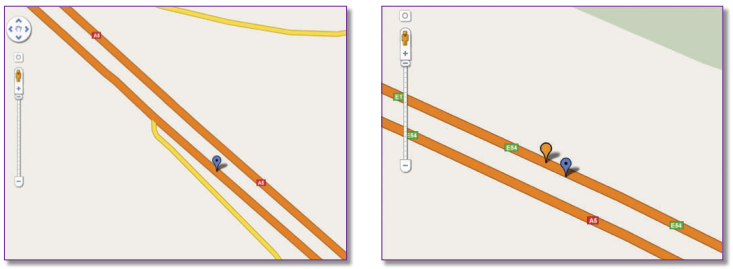
Pour le reste, le travail d'analyse a également porté sur la façon d'importer et de lire les données d'infos-trafic. Le standard *Datex* propose deux mécanismes d'échanges : *push* à l'initiative du fournisseur et *pull* à l'initiative du client. Actuellement, la seule implémentation disponible sur le site *Bison-Futé* donne accès à une ressource XML déposée sur un serveur HTTP et régénérée toutes les quarts d'heure. L'application MFC intégrant les composants *Loxane X* doit donc télécharger par une requête HTTP les données XML, puis les analyser pour ne retenir que les informations utiles listées ci-dessus. Le processus complet, regroupant les trois opérations (téléchargement, interprétation, écriture du fichier) est exécuté en tâche de fond dans un processus dédié, initialisé au démarrage de l'application. Celui-ci est mis en sommeil et réactivé automatiquement selon une fréquence paramétrable (quinze minutes par défaut). À noter qu'en raison de la taille du fichier (plusieurs centaines de milliers de lignes), le processus ne peut être invoqué trop fréquemment. L'interprétation du fichier XML repose sur la

```
</ns1:situationRecord>
<ns1:situationRecord id="X319806_2" xsi:type="ns1:MaintenanceWo
 <ns1:situationRecordCreationReference>2</ns1:situationRecordCr
 <ns1:situationRecordCreationTime>2009-06-11T15:00:00+02:00</ns
 <ns1:situationRecordVersion>2</ns1:situationRecordVersion>
 <ns1:situationRecordVersionTime>2009-06-11T15:00:00+02:00</ns1
 <ns1:situationRecordFirstSupplierVersionTime>2009-06-11T15:00:
 <ns1:probabilityOfOccurrence>probable</ns1:probabilityOfOccurr
 <ns1:sourceInformation>
  <ns1:sourceName>
   <ns1:value>fr202</ns1:value>
  </ns1:sourceName>
 </ns1:sourceInformation>
 <ns1:validity>
  <ns1:validityStatus>active</ns1:validityStatus>
  <ns1:validityTimeSpecification>
   <ns1:overallStartTime>2009-06-19T15:30:00+02:00</ns1:overall
   <ns1:overallEndTime>2009-06-19T16:00:00+02:00</ns1:overallEn
  </ns1:validityTimeSpecification>
 </ns1:validity>
 <ns1:impact>
  <ns1:impactDetails>
   <ns1:capacityRemaining>50.0</ns1:capacityRemaining>
  </ns1:impactDetails>
  <ns1:delays/>

 </ns1:impact>
 <ns1:generalPublicComment>
  <ns1:comment>
   <ns1:value>Nature    PONTAGE</ns1:value>
  </ns1:comment>
 </ns1:generalPublicComment>
 <ns1:groupOfLocations>
  <ns1:locationContainedInGroup xsi:type="ns1:Linear">
   <ns1:tpeglinearLocation>
    <ns1:tpegDirection>unknown</ns1:tpegDirection>
    <ns1:tpegLocationType>segment</ns1:tpegLocationType>
    <ns1:to xsi:type="ns1:TPEGNonJunctionPoint">
     <ns1:pointCoordinates>
      <ns1:latitude>4.95292</ns1:latitude>
      <ns1:longitude>48.05591</ns1:longitude>
     </ns1:pointCoordinates>
```

Figure 10 – Extraits du fichier XML Bison-Futé s'appuyant sur le schéma Datex2 ; extrait du fichier interne d'archivage des évènements.

Figure 11 – Appariement des données d'infos-trafic avec une base cartographique.



bibliothèque C++ *Xerces* qui utilise l'API DOM *(Document Object Model)* pour manipuler le fichier : cette API permet entre autres avec la méthode *getElementsByTagName* de stocker dans des listes (*DomNodeList*) le contenu de toutes les balises de même nom. Dans le cadre du projet, seul le service *Bison-Futé* a été exploité ; toutefois, on pourrait selon le même principe se connecter à d'autres services HTTP, chaque source serait alors traitée par un processus différent, générant un fichier d'évènements dédié, chacun des fichiers étant en suite intégré à une base de données unique.

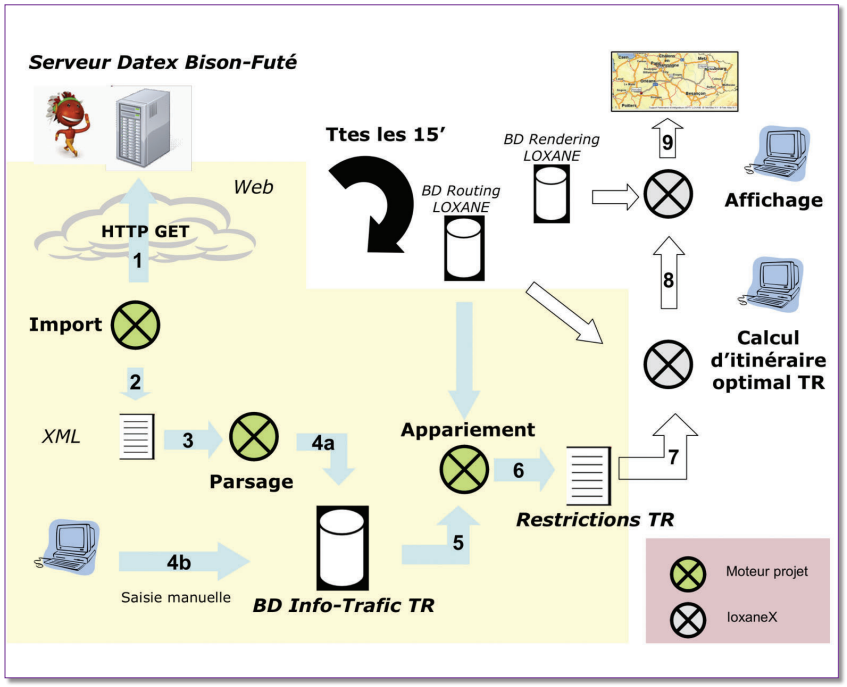


Figure 12 – Architecture de la solution proposée par l'équipe projet ASIG ENSG.

En résumé, le projet a proposé de développer deux nouveaux moteurs de calculs : un moteur d'import temps-réel d'infos-trafic multi-sources produisant des données fichiers temps-réel dans un format interne et un moteur de calcul d'une base de données de *restrictions* exploitant ces données fichier. Actuellement, pour certains développements spécifiques, des compléments sont déjà fournis aux intégrateurs sous forme de bibliothèques additionnelles. Ces deux moteurs pourraient donc être diffusés de la même manière (Figure 12).

Conception et développement du prototype

La réalisation du prototype s'est faite dans un laps de temps très court (trois semaines). Les étudiants ont procédé en trois temps : définition collégiale du format interne d'archivage des évènements, des fonctionnalités (Figure 13) et de l'interface homme-machine du prototype de validation ; puis, conception et développement en binôme des moteurs d'import et de calcul des restrictions ; enfin branchement collégial des moteurs sur le prototype.

Le prototype conçu permet à l'utilisateur de définir le mode de connexion aux données d'infos-trafic : automatique (téléchargement HTTP GET toutes les quinze minutes) ou bien à la demande. Des champs de saisie donnent également la possibilité d'entrer manuellement de l'info-trafic. Cette fonctionnalité a été ajoutée uniquement pour montrer la capacité de la solution à prendre en compte une source autre que *Bison-Futé*. L'appariement et l'archivage interne des restrictions sont déclenchés par bouton commande. *RouteFinder* et *MapDisplay* sont

utilisés pour représenter en mode tableur ou cartographique les évènements temps réel archivés. Il est aussi possible d'obtenir un fichier texte recensant, pour une route désignée par son numéro, toutes les perturbations actives. Enfin, on peut, bien sûr, calculer, sur le réseau routier national, un itinéraire tenant compte des infos-trafic gérées par le système.

Pour la partie développement, le groupe *« Import »* a fait appel à des bibliothèques C++ externes : *curllib* pour l'envoi de requête HTTP et xerces-c_3 pour l'analyse du document XML ; il a utilisé des bibliothèques internes *Windows*™ pour la programmation du processus et l'écriture du fichier d'évènements temps réel. Les développements ont été distribués sous forme de deux classes C++ (*tools* et *evenement*).

Le groupe *« Restriction »* a, quant à lui, fait appel à la fois aux APIs des composants *Loxane X* ainsi qu'à plusieurs bibliothèques annexes *Loxane* ; les développements concernant l'appariement et l'archivage des fichiers de *restriction* sont livrés sous forme d'une classe C++ (*segment*) ; chacun des deux groupes a validé ses classes par le biais de tests unitaires.

Le binôme « *restriction* » a par ailleurs étudié très sérieusement le format de données propriétaire de graphe ; la mise au point de l'algorithme d'appariement repose sur la compréhension du rôle joué par les *KeyPoints* ou nœuds du réseau ; ces derniers correspondent à des points remarquables (carrefours, endroits matérialisant un changement de valeur d'attribut de la route) qui décrivent de manière généralisée le réseau routier. Ce sont eux que le moteur d'appariement cherche à identifier avec les extrémités des segments *Bison-Futé*. Ce sont également eux qui sont utilisés pour générer la géométrie des objets *restriction*.

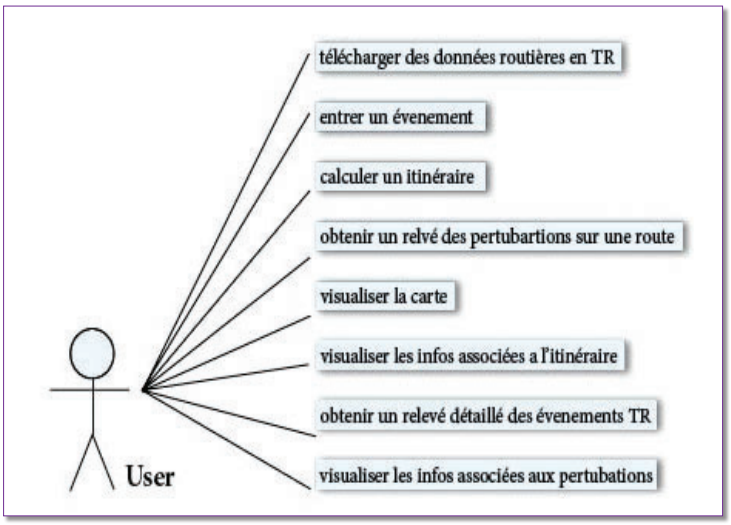


Figure 13 – Diagramme de cas d'utilisation du prototype mis en oeuvre pour valider la proposition.

L'algorithme proposé est volontairement simpliste : il fallait trouver un compromis permettant à l'équipe de s'interfacer à la date convenue. En substance, on recherche pour les nœuds initial et final de chaque segment de *restriction* le segment routier (morceau de route entre deux *KeyPoints*) *Loxane* le plus proche (on restreint la recherche à un voisinage). On compare ensuite les segments *Loxane* de *début* et de *fin* : s'ils sont identiques (même identifiant interne), et s'ils portent le même numéro de route que la *restriction Bison-Futé*, alors on crée un objet *restriction* à partir des deux segments et des segments intermédiaires. Si les conditions précédemment énumérées ne sont pas vérifiées, alors et uniquement si l'un des deux segments porte le même numéro que le segment *Bison-Futé*, on crée pour chacun un objet *restriction* local en le dupliquant. La figure 15 montre les deux cas : une *restriction* continue de trafic sur l'A71 dans le sens Saint-Amand-Montrond–Bourges (ruban vert) ; des restrictions localisées sur la A20 (segments

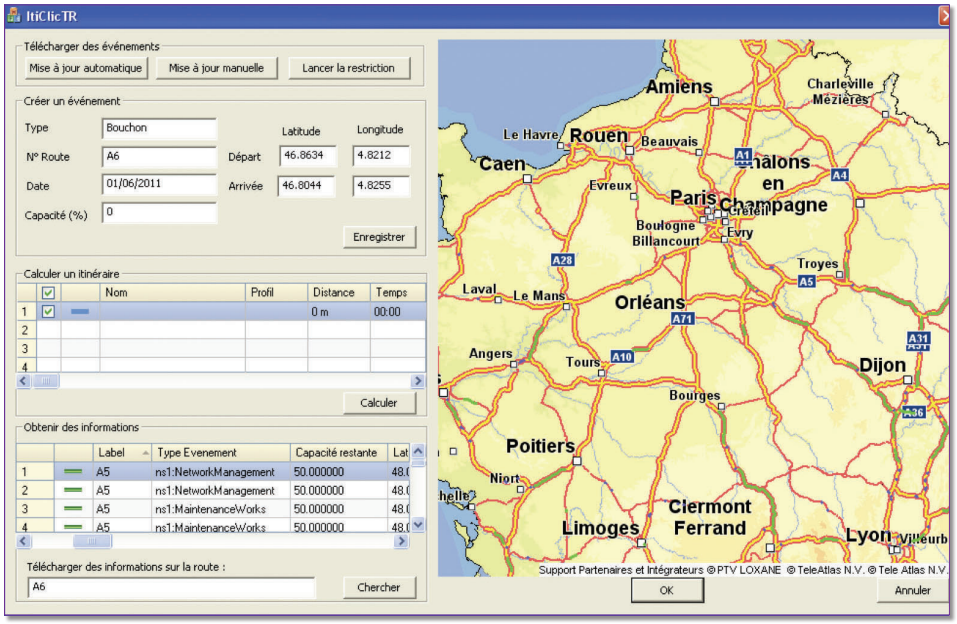


Figure 14 – Interface Homme-Machine du prototype de validation.

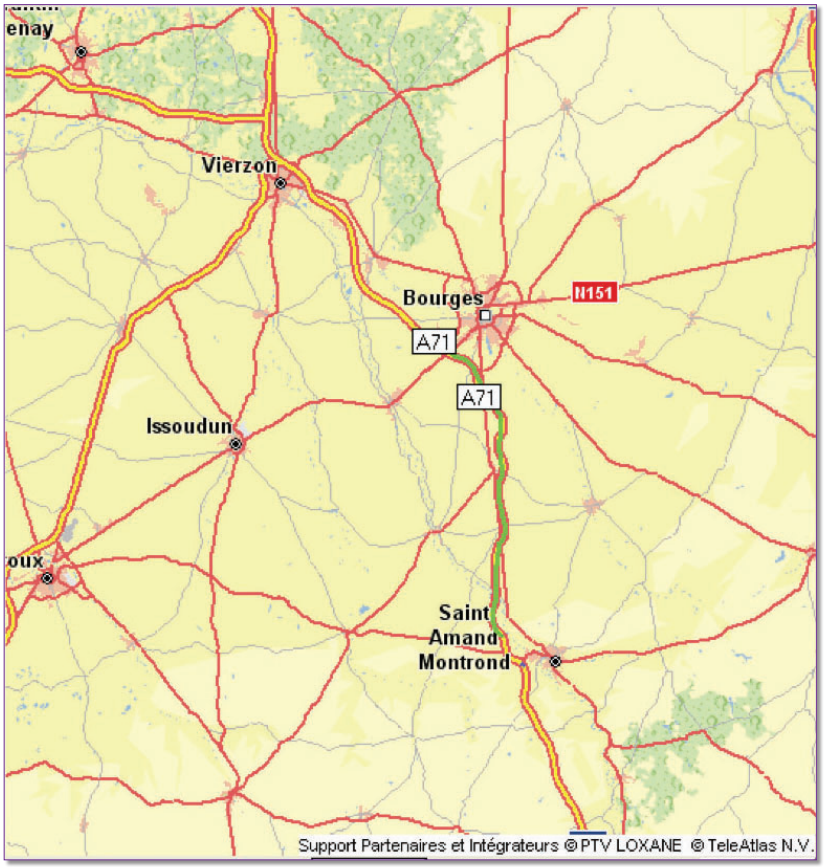
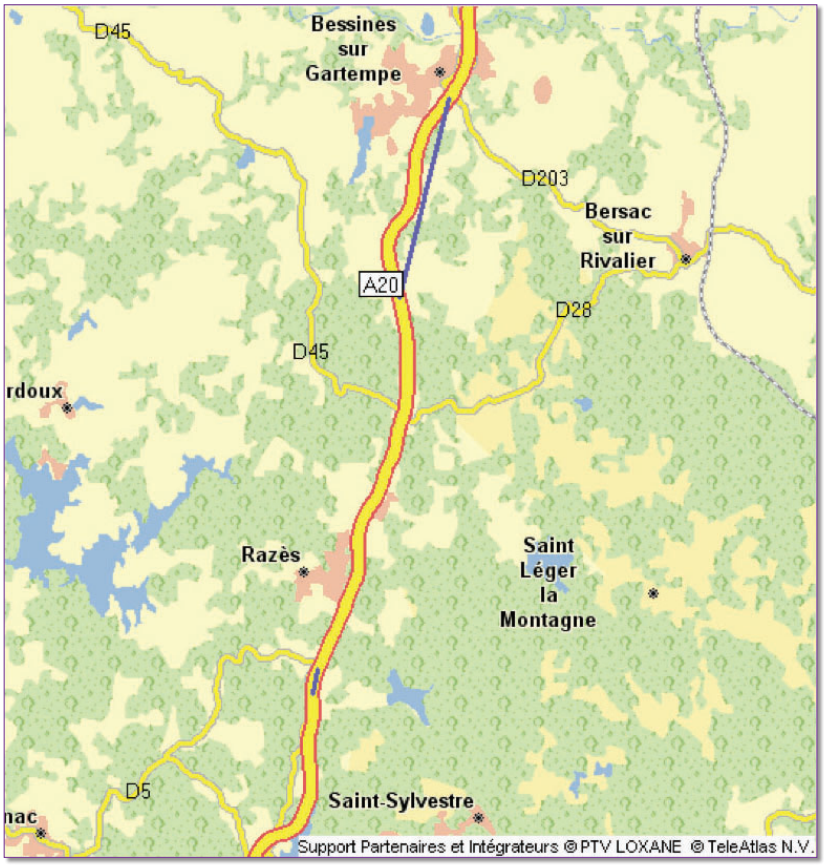


Figure 15 – Différents cas d'appariement.



bleus). À noter que seul le graphe des autoroutes a été exploité pour l'appariement.

En l'état, sur 956 évènements autoroutiers recensés dans le fichier XML source, 370 sont appariés de manière satisfaisante soit environ 40 %.

Conclusion

Les étudiants ASIG de la promotion 2010-2011 étaient confrontés à un exercice difficile : ils ont su mobiliser leurs connaissances, mais surtout ils ont su s'organiser de manière efficace pour aboutir dans le laps de temps très court imparti.

Les objectifs pédagogiques sont pleinement atteints : chacun a montré son aptitude à travailler dans un contexte professionnel (définition d'objectifs, conception détaillée, mise en œuvre, respect des délais), et, ce qui ne saurait être rangé au rang de l'anecdote, leur travail valide sans ambiguïté l'idée d'utiliser le mécanisme des restrictions pour intégrer les infos-trafic temps réel dans *Loxane X*. Ce résultat est très prometteur et gageons que le jury du trophée étudiant de l'intégration 2011 n'y sera pas insensible.

Pour information, ce projet industriel a été introduit à la rentrée 2010 dans le programme d'enseignement de la filière ASIG du cycle ingénieur ENSG : il sera bien entendu reconduit à la rentrée prochaine, avec sans doute deux sujets puisque la promotion comptera une dizaine d'étudiants.

Si vous souhaitez travailler avec eux (appel à idées pour de nouveaux projets) ou bien si vous souhaitez en savoir plus sur cette formation, vous pouvez vous rendre sur le site de l'école (http://www.ensg.eu, rubrique *Formation 3ème année ingénieur*) ou encore me contacter directement. ■