École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
5e année
2013 - 2014

Rapport de PFE

MobiRev :

La Mobilité Révélée par GPS

Encadrant
Carl ESSWEIN
carl.esswein@univ-tours.fr

Étudiant
Maxime SERRA
maxime.serra@etu.univ-tours.fr

Maîtrise d'ouvrage
Benoît FEILDEL
benoit.feildel@univ-tours.fr

DI5 2013 - 2014

Université François-Rabelais, Tours

Version du 4 mai 2014

Table des matières

Table des figures

Remerciements

Par cette page je souhaiterais apporter mes remerciements aux personnes suivantes :

Je remercie tout d'abord Carl ESSWEIN, mon encadrant pour sa présence et ses conseils avisés tout au long de l'année.

Je remercie également Benoît FEILDEL pour sa disponibilité lorsqu'il me fallait des renseignements ou bien des rendez-vous ainsi que pour avoir joué le jeu en tant que maîtrise d'ouvrage.

Enfin je remercie Claire LAFITTE, étudiante en $5^{ème}$ année au département Aménagement de Polytech Tours, pour m'avoir apporté son soutien et sa relecture de mon rapport.

Introduction

Le Projet de Fin d'Etude, ou P.F.E, est la partie terminale du cycle ingénieur à Polytech Tours. Ce projet s'effectue sur l'intégralité de la dernière année d'étude et permet de mettre en pratique les compétences à la fois techniques et fonctionnelles acquises lors du cursus.

Ainsi, le P.F.E est donc l'occasion pour les étudiants de démontrer leur aptitude à mener à bien un projet d'une certaine envergure en maîtrisant toutes les étapes d'audit, de conception et de réalisation.

Le sujet du projet de fin d'étude que j'ai choisi est le suivant : **"MobiRev : La Mobilité Révélée par GPS"**. Ce dernier s'est fait en collaboration avec un enseignant du Département Aménagement de Polytech Tours, **Benoît FEILDEL** qui est également un chercheur à l'UMR 7324 CITERES[1] de l'université de Tours.

L'objectif de ce document va être dans un premier temps d'expliquer plus précisément en quoi consiste le projet et notamment le contour scientifique de ce dernier afin d'en exposer les objectifs. Après quoi je détaillerai les différents choix opérés en termes de technologie et de conception avant de rentrer dans la phase de développement du projet.

1. Créée en 2004, l'Unité Mixte de Recherche CITERES (**CI**tés, **TER**ritoires, **E**nvironnement et **S**ociétés) est venue renforcer et structurer le potentiel de recherche de l'Université de Tours sur la thématique "Villes et Territoires". Son objectif est d'analyser les dynamiques spatiales et territoriales des sociétés. (http://citeres.univ-tours.fr/spip.php?rubrique82)

Présentation du projet

2.1 Contexte

Afin de comprendre le contexte du PFE, il est nécessaire de remonter jusqu'en 2009. Cette année-là, Le PUCA [1], qui est affilié au Ministère de l'Écologie du Développement durable et de l'Énergie, lance un appel à projet sur le thème suivant : *"La mobilité et le périurbain à l'impératif de la ville durable. Ménager les territoires de vie des périurbains"*.

C'est en réponse à cet appel à projet que l'UMR CITERES de l'université de Tours propose de mener l'enquête PériVia [2] durant laquelle une quarantaine d'individus se sont portés volontaires pour être équipés de dispositifs GPS qui ont enregistré tous leurs trajets pendant plusieurs semaines. Une partie de ces données a ensuite été analysée et traitée manuellement afin d'en extraire diverses informations pertinentes pour les recherches des enquêteurs.

De la même manière, en 2011, c'est la Région Centre qui a lancé un appel à projet portant sur le thème des *"Nouvelles mobilités et urbanisme rural dans les espaces à faible densité"*. Là encore, l'UMR CITERES de Tours a répondu à cet appel à projet et a dans ce cadre mené une nouvelle enquête, MOUR [3]. Dans le cadre de cette enquête, ce sont cette fois une vingtaine de personnes qui ont accepté de s'équiper d'un dispositif GPS pour enregistrer tous leurs trajets quotidiens.

Dû à la grande masse d'informations récoltées et au traitement manuel sur ces dernières, leur analyse ne s'est seulement faite que sur la moitié des personnes concernées par l'enquête. Il en va de même pour l'enquête MOUR dont l'analyse et le dépouillement des résultats représentent un travail colossal.

1. **P**lan **U**rbanisme **C**onstruction **A**rchitecture : Le PUCA développe des programmes de recherche incitative, des actions d'expérimentations et apporte son soutien à l'innovation et à la valorisation scientifique et technique dans les domaines de l'aménagement des territoires, de l'habitat, de la construction et de la conception architecturale et urbaine.(http://rp.urbanisme.equipement.gouv.fr/puca/puca/presentation.htm)

2. Le **Péri**urbain à l'épreuve des modèles d'habiter. La **Via**bilité périurbaine entre théorie(s) et pratique(s).

3. **Mo**bilité et **U**rbanisme Rural

2.2 Objectifs

Comme exposé précédemment, l'UMR CITERES a donc récolté au fil du temps d'importantes masses de données GPS sous la forme de fichiers plats, portant l'extension ".gpx". Les objectifs attendus à l'issue de ce projet de fin d'étude sont donc multiples :

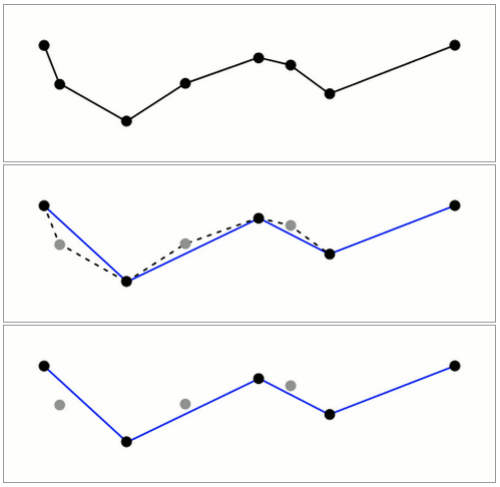
- Il s'agit dans un premier temps d'avoir un système (logiciel) qui soit capable de prendre en entrée les fichiers récoltés lors des différentes enquêtes des chercheurs de l'UMR CITERES.

- Après avoir pris en entrée les fichiers de données GPS, l'objectif suivant est de pouvoir appliquer des algorithmes sur ces données afin de les "nettoyer". Ceci doit consister à corriger les données GPS jugées aberrantes dûes notament à la précision approximative des appareils GPS. Parmi les corrections à effectuer, ces dernières peuvent être catégorisées dans les algorithmes suivants, qui seront explicités plus en détail dans une partie ultérieure de ce rapport :

  — Réduction du nombre de points

  L'algorithme en charge de ce traitement doit être capable de prendre un ensemble de points GPS en entrée (i.e. une trace GPS [4]) et d'en réduire le nombre, selon un seuil paramétrable, tout en conservant l'allure de la trace sans en dénaturer les informations pouvant en être extraites comme par exemple les routes empruntées.

  Comme illustré sur le schéma ci-contre, il s'agit se simplifier une polyligne en assimilant, quand c'est possible, certaines courbes à une droite.

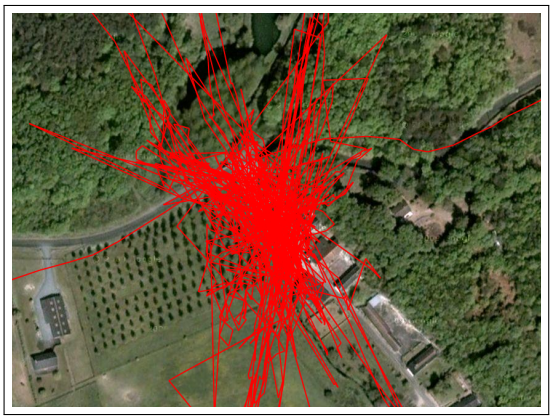


  — Suppression des points aberrants
  L'algorithme de suppression des points aberrants doit être capable, pour une trace donnée en entrée, d'en éliminer les points qui seraient humainement jugés "aberrants".
  Pour se faire, l'algorithme devra se servir des données de vitesse instantanée, de vitesse moyenne, de coefficient d'accélération ou de toutes autres données qui pourraient être nécessaires à la suppression de ces points aberrants.

4. On qualifiera de *trace GPS* (ou *trace*) une suite de coordonnées, latitudes et longitudes, sur une période donnée accompagnée éventuellement d'autres informations complémentaires telles que la vitesse instantanée, la date, l'altitude, etc.

— Traitement des points d'arrêts

La détection des points d'arrêts peut être vu comme un cas particulier des points aberrants. En effet, lorsqu'un dispositif GPS se trouve à l'intérieur d'un bâtiment, les données reçues sont moins précises. Cela entraine des données très erratiques tant sur leur forme que sur leur donnée physique telle que le coefficient d'accélération par exemple.
De plus, lorsqu'un dispositif GPS demeure dans ces conditions, l'accumulation de tous ces points aberrants constitue un amas de données illisibles et incohérentes. D'où la nécessité de détecter ces points d'arrêts afin de les remplacer. Ci-dessous un exemple de "pelotte" qui constitue un point d'arrêt.



- Par la suite, après l'importation et le traitement de ces données GPS, l'objectif suivant est d'en extraire des informations à fournir aux chercheurs. Ces informations peuvent être triviales telles que des courbes de vitesse ou d'accélération, mais elles peuvent aussi s'avérer complexes comme par exemple inférer sur le mode transport utilisé ou encore calculer les émisions de gaz à effet de serre.

- Enfin, le dernier objectif est bien entendu de pouvoir sauvegarder son travail et donc d'exporter les données traitées dans un fichier de même format.

Pour prendre connaissance de l'ensemble des fonctionnalités plus détaillées qui découlent des objectifs présentés ci-dessus, il est nécessaire de se référer au cahier de spécification du projet.

2.3 Démarche projet

2.3.1 Méthode

Lors du commencement du projet, comme pour tout projet, il a d'abord fallu déterminer quelle méthode de gestion de projet allait être employée parmi toutes celles que nous avions eu l'occasion de voir en cours : AGILE, ITIL, PRINCE, cycle en V, cycle en spiral etc.

Finalement, de par la disponibilité de la maîtrise d'ouvrage, il a été décidé d'adopter une démarche projet plutôt orientée AGILE. En effet, il était relativement aisé d'avoir un rendez-vous avec Benoît FEILDEL. Ainsi, cela a donc facilité les différents retours notamment lors des phases d'audit, de maquettage ou encore de recette.

2.3.1.1 Notion de Sprint

Dans la cadre d'une gestion AGILE d'un projet de développement logiciel, il est nécessaire d'aborder la notion de **sprint**. En effet, le sprint est un composant de la méthode Scrum qui fait partie intégrante de l'agilité.

Par nature, un sprint est défini sur une période allant de deux à cinq semaines maximum. Pendant cette période, un certain nombre d'objectifs, sous forme de liste de tâches, sont fixés. Du fait de ces courtes périodes de temps, la complexité des différents objectifs à atteindre est donc moindre. C'est le principe du *"Diviser pour régner"*.

Dans le cadre du projet de fin d'étude, il a été décidé avec mon encadrant Carl ESSWEIN que ces sprints seraient d'une durée de un mois, soit quatre semaines. Des points réguliers étaient donc fais avec lui afin de mesurer l'avancement. A chaque début de mois, l'un de ces points était consacré à la rétrospective du sprint précédent et à la définition des objectifs du sprint des semaines à venir sous forme d'une liste de tâches.

2.3.2 Outils

Pour m'accompagner dans le déroulement du PFE, je me suis doté de deux outils indispensables dans la gestion de projet de développement logiciel, Redmine et Subversion (i.e. SVN).

Afin de faciliter le suivi des projets de fin d'étude, Polytech Tours met à la disposition de ses étudiants un accès au serveur de l'école sur lequel est installé et paramétré l'outil Redmine. Grâce à cet outil de gestion de projet mon encadrant et moi-même avons pu suivre progressivement l'avancement du projet notamment en termes de livrable et de deadline des différents objectifs.

Par ce biais, les différents documents relatifs au projet (cahier des charge, cahier de spécification, maquettes, etc.) y étaient déposés. De même, les deadlines et demandes de tâches étaient intégrées au calendrier de Redmine afin d'avoir une vue d'ensemble de ce qui était fait et de ce qui restait à faire.

En complément de l'outil Redmine, le logiciel Subversion (i.e. SVN) y était directement couplé. SVN est un logiciel dit de "versionning". Il s'agit d'un logiciel de contrôle de version de sources. Le dépot SVN qui était configuré avec Redmine m'a donc permis d'y sauvegarder incrémentalement, avec possibilité de retour arrière, les sources du logiciel que j'ai développé pour mon PFE.

Phase d'étude

3.1 Audit de la maîtrise d'ouvrage

3.1.1 Recueil du besoin

Comme mentionné précédemment, la maîtrise d'ouvrage, en la personne de Benoît FEILDEL, pouvait très facilement se rendre disponible afin de participer à des interviews avec mon encadrant Carl ESSWEIN et moi-même. Ainsi, l'audit de la maîtrise d'ouvrage a pu rapidement s'effectuer lors des premières semaines du début du projet.

Le recueil du besoin a été la toute première étape du projet de fin d'étude. Nous nous sommes réunis afin que Benoît FEILDEL puisse nous exposer clairement la problématique du projet ainsi que les besoins sous-jacents.

Les différentes sessions ayant eu lieu pour recueillir le besoin m'ont permis dans un premier temps de mettre au propre le **Cahier des Charges** (voir section 7.3). Ce cahier des charges a ensuite été pour moi la base sur laquelle le **Cahier des Spécifications** (voir section 7.4) a pu être rédigé.

Après plusieurs allers-retours, ces deux documents ont été tour à tour validés par la maîtrise d'ouvrage ainsi que par mon encadrant. Ces navettes de validation ont permis de se mettre d'accord d'une part sur le découpage des fonctionnalités souhaitées mais également sur une version plus ou moins stable de la maquette de l'application.

C'est ainsi que le recueil du besoin a pu lentement converger vers une vision plus claire et précise des fonctionnalités souhaitées.

3.1.2 Maquettage

Au cours des différents entretiens et après s'être mis d'accord quant à la nature de l'application à développer, diverses versions de maquettes ont été élaborées.

Une **première maquette** (voir section 7.1.1) de ma conception a été présentée à la maîtrise d'ouvrage basée sur une vision initiale que je m'étais fait de l'application, connaissant ses fonctionnalités.

A partir de cette dernière, Benoît FEILDEL a pu y apporter sa vision des choses en suggérant des axes d'améliorations qui ont permis d'arriver à une **version stable** (voir section 7.1.2) de la maquette de l'application.

3.2 Étude de l'existant

3.2.1 ItiRestitution

Le laboratoire "Image, Ville, Environnement" de l'UMR 7362 du CNRS et de l'Université de Strasbourg a travaillé sur un projet visant à traiter des traces GPS en vue de restituer un cheminement urbain cohérent à partir de données GPS brutes.

Dans ce cadre, il a été développé un logiciel baptisé *ItiRestitution*. Ce dernier comporte plusieurs procédures qui permettent de traiter des données GPS afin d'en identifier principalement les itinéraires et les points d'arrêt, comme son nom l'indique.

C'est ce logiciel qui a inspiré la maîtrise d'ouvrage pour le projet de fin d'étude MobiRev et en particulier pour ce qui touche à la détection des points d'arrêt ainsi qu'à la représentation visuelle de ces derniers. Cependant, comme l'a mentionné la maîtrise d'ouvrage, elle désirait une solution "maison" qui soit plus personnalisée à ses besoins.

3.2.2 Gpx Editor

Gpx Editor[1] est une autre solution existante qui permet d'importer des fichiers issus de dispositifs GPS. Cependant, ce logiciel-là est plus orienté vers la manipulation des traces proprement dites comme le fait de fusionner des fichiers ou encore couper des traces.

Il permet néanmoins d'appliquer quelques traitements sur les traces comme la réduction du nombre de points[2] par exemple. Il y a également une fonctionnalité de détection de pelottes, c'est-à-dire des points d'arrêt, mais qui ne s'est pas avérée très concluante sur les différents fichiers testés.

Malgré tout, la présentation de ce logiciel à la maîtrise d'ouvrage a permis de faire naître de nouvelles idées notamment en ce qui concerne l'ergonomie de l'application.

1. Site Web : http://sourceforge.net/projects/gpxeditor/

2. Voir section 3.5.1

3.3 Définition du contour technologique

3.3.1 Type d'application

La question du type d'application à élaborer s'est posée dès l'aurore du projet. Il était en effet nécessaire de savoir dans quelle direction partir. Ainsi, une interview avec Benoît FEILDEL a été réalisée en ce sens.

La maîtrise d'ouvrage n'avait pas encore de vision précise et fermée sur le sujet, j'ai donc dans un premier temps suggéré le principe d'une application web, afin d'en favoriser l'accessibilité en limitant les dépendances sur les postes client. Cependant, la question de l'hébergement de l'application posait problème.

En creusant un peu d'avantage, il s'est avéré que la maîtrise d'ouvrage n'avait pas de réelle contrainte quant à la sécurité en termes de gestion des rôles et des autorisations et de surcroît préférait au final avoir un client lourd.

La nature de l'outil à élaborer serait donc une application hors-ligne mono-poste et mono-utilisateur. C'est-à-dire un programme exécutable classique mais devant néanmoins permettre des connexions à internet, selon les fonctions à utiliser.

À moyen ou long terme, l'idée et la possibilité de faire évoluer l'outil en une application web n'étaient pas complètement fermées. Cela impliquait donc que l'application soit construite de manière très modulaire afin d'en faciliter la possible maintenance évolutive des années à venir.

3.3.2 Technologie d'implémentation

Afin de concevoir l'application désirée, il a fallu réfléchir sur la technologie d'implémentation à utiliser. Le choix de cette technologie à employer s'est porté sur le langage **C# avec l'utilisation du framework .NET**.

Le choix d'utiliser la dite technologie pour réaliser cette application n'est pas le fruit du hasard et je l'ai motivé par les raisons suivantes :

- L'utilisation du framework .NET avec le langage C# est par nature très modulaire, ce qui nous convient très bien ici.
- Les applications logicielles construites avec le framework .NET sont aisément convertibles en une application web. Simplement, seule la partie visuelle est à adapter et la partie fonctionnelle ne nécessite pas ou peu de retouches.
- Le langage C# (avec le framework .NET) étant très populaire, de nombreuses librairies et API sont disponibles pour convenir au besoin de l'outil à développer.

- Au sein de Polytech Tours, de nombreux PFE qui ont été proposés cette année employaient cette technologie d'implémentation. Ce qui peut laisser présager que des compétences dans ce domaine existeront dans le futur en cas de maintenance de l'outil.

- Enfin, cette technologie est bien connue et maitrisée de la maitrise d'œuvre (moi-même) ce qui permettra de garantir la robustesse et la pérennité souhaitées par la maîtrise d'ouvrage (Benoît FEILDEL).

3.4 Recherche et Conception

3.4.1 Etude des librairies SIG

Afin de pouvoir intégrer dans l'application une vue géographique des traces GPS importées par l'utilisateur, il a été nécessaire de rechercher une librairie de type "SIG" [3] permettant de faire cela avec la technologie d'implémentation sélectionnée.

Un état de l'art sur ce type de librairie a rapidement montré un nombre conséquent de résultats. Cependant, après approfondissement, quelques bibliothèques potentielles sont sorties du lot et le choix s'est ensuite fait sur la base de la matrice de décision suivante réunissant les principales fonctionnalités requises pour le projet :

		Librairies		
		GoogleMap API	OsmSharp	Gmap.NET
Fonctionnalités	Implémentation C#	✗	✓	✓
	Routes	✓	✓	✓
	Calques	✓	✓	✓
	Marqueurs	✓	✓	✓
	Fonds cartographiques de multiples fournisseurs	✗	✗	✓
	Vue Satellitaire	✓	✗	✓
	Facilité d'intégration dans un Winform	✗	!	✓

Matrice de décision des librairies à utiliser.

L'API fourni par GoogleMap est très complète et offre un nombre impressionnant de possibilités. Ces dernières permettent de répondre en grande partie aux besoins du projet. Cependant, cette API est écrite en Javascript ce qui rend difficile son intégration dans une application Windows de type "client-lourd". Par ailleurs, la possibilité d'utiliser différents fonds cartographiques est restreinte à ceux de Google.

Le projet communautaire OpenStreetMap fournit également une API pour accéder à sa base de données et effectuer un certain nombre d'opérations. Il existe même une implémentation de cette API en langage C# baptisée *OsmSharp*. Néanmoins, cette bibliothèque est plus orientée vers le calcul de route que la représentation visuelle des données GPS.

La librairie *Gmap.NET* [4], raccourci pour **G**reat **Maps .NET**, est une librairie gratuite et open source s'appuyant sur le framework .NET et fournissant une couche d'abstraction permettant d'utiliser tous les services de cartes tels que GoogleMap, OpenStreetMap, BingMap et bien d'autres. Ce qui permet d'accéder aux fonctions de calcul d'itinéraires de GoogleMap et OpenStreetMap.

Il est notamment possible de récupérer les fonds de cartes de tous ces fournisseurs et de les utiliser pour afficher un rendu de coordonnées GPS au sein d'une application Windows. Toute la partie graphique étant largement paramétrable, aussi bien la couleur du tracé et son épaisseur, que les marqueurs de départ et d'arrivée, etc.

3. SIG : **S**ystème d'**I**nformation **G**éographique
4. Site de la bibliotèque GMap.NET : http://greatmaps.codeplex.com

De surcroît, l'aspect open source de cette bibliothèque s'est avéré utile dans la mesure où j'ai pu en récupérer le code source afin notamment d'ajouter un mode de visualisation des traces sans fond cartographique pour répondre à une demande du client : avoir une visualisation "brute" d'une trace.

3.4.2 Modélisation UML

La modélisation UML de l'application et plus particulièrement la constitution d'un diagramme des classes métier a été une partie conséquente et essentielle dans le déroulement du projet.

L'enjeu était ici d'obtenir un diagramme métier qui soit le plus mature et stable possible puisque toute l'architecture de l'application s'appuie dessus. La facilité d'évolution et de maintenance y étant fortement liée.

La mise au point du **diagramme des classes métier** (voir section 7.2) a ainsi occupé la majeure partie du premier semestre du projet. Au final, c'est plus d'une vingtaine de classes qui le composent. Il a été travaillé afin d'être le plus faiblement couplé possible pour permettre une évolution facilitée dans le futur si le projet est repris en vue d'y ajouter des fonctionnalités.

3.4.2.1 Entrées/Sorties

Afin d'établir le diagramme des classes, il a d'abord été nécessaire d'analyser le domaine fonctionnel. Lorsque l'on parle de données GPS ou bien de traces GPS, cela implique dans un premier temps de s'intéresser au format de fichier GPX [5]. Dans le cadre de la gestion des entrées et sorties, il a donc fallu des classes permettant de représenter ce format.

Le contenu d'un fichier portant l'extension GPX est en fait du langage XML. On y trouve donc un ensemble de balises qui sont organisées et structurées selon le DTD [6] du format GPX.

Parmi ces balises, on trouve en premier lieu la balise <**trk**> pour **track** ou *trace* en français. Hiérarchiquement de plus haut niveau, elle va contenir un ou plusieurs *segments*. Ces segments sont représentés par les balises <**trkseg**>. Ces dernières représentent une suite continue dans le temps de points GPS. Si le dispositif GPS avait une interruption de signal pour une quelconque raison, un nouveau segment serait alors créé lorsque le signal serait retrouvé.

Les points que l'on trouve dans un segment sont représentés par la balise <**trkpt**> . C'est cette dernière qui contient toutes les informations relatives à un point GPS : la latitude, la longitude, la vitesse instantanée, la date et l'altitude.

5. GPX : **G**PS e**X**change **F**ormat est un format de fichier en XML qui permet de décrire une suite de coordonnées GPS avec un certain nombre de méta-données telles que la vitesse, l'altitude, la date, etc.

6. DTD : **D**ocument **T**ype **D**efinition ou **D**éfinition de **T**ype de **D**ocument, correspond à la description des règles syntaxiques et hiérarchiques quant aux balises que l'on est censé trouver dans un document XML.
DTD du format GPX : http://www.topografix.com/gpx/1/0/gpx.xsd

Ce sont ces trois principales composantes qu'il a fallu modéliser afin d'intégrer la notion d'entrées/sorties dans la diagramme UML. Cela a donné naissance aux classes suivantes :

- GPXFile : pour représenter un fichier GPX
- GPXTrack : représentant une balise <**trk**> avec une description éventuelle et une collection des segments associés
- GPXSegment : représentant une balise <**trkseg**> avec une liste des points associés
- GPXPoint : représentant une balise <**trkpt**> avec toutes les informations mentionnées précédemment

3.4.2.2 Classes métier

Les classes créées précédemment permettent de manipuler toutes les notions élémentaires liées au domaine GPS. Cependant, au-delà de l'aspect des entrées/sorties et à la vue des fonctionnalités à implémenter pour manipuler et traiter les traces, il est rapidement devenu évident qu'il allait y avoir besoin de notions plus complexes qui allaient se décoller du modèle GPX de base.

Notion de POI

Dans un premier temps, il a fallu revoir la notion de *point GPS*. Si la classe *GPXPoint* contient déjà les informations les plus pertinentes concernant un point, la notion de *point d'arrêt* ne peut pas être correctement représentée pour des raisons sémantiques. En effet, un point d'arrêt peut être potentiellement constitué d'une multitude de points GPS.

On préfèrera ainsi la notion de **POI** pour *"point of interest"* ou *point d'intérêt* en français. Ce dernier représente donc un point mais à un niveau plus abstrait. Un *POI* pourra donc représenter soit un simple point GPS, soit être un point d'arrêt. Un point d'arrêt étant constitué de plusieurs *POI*.

Les classes *GPXPoint* et *POI* partageant un socle commun mais avec malgré tout des sémentiques différentes, il était tout naturel qu'elles aient une classe mère commune *PointGPS* dont elles sont les spécialisations.

Notion de Trajet

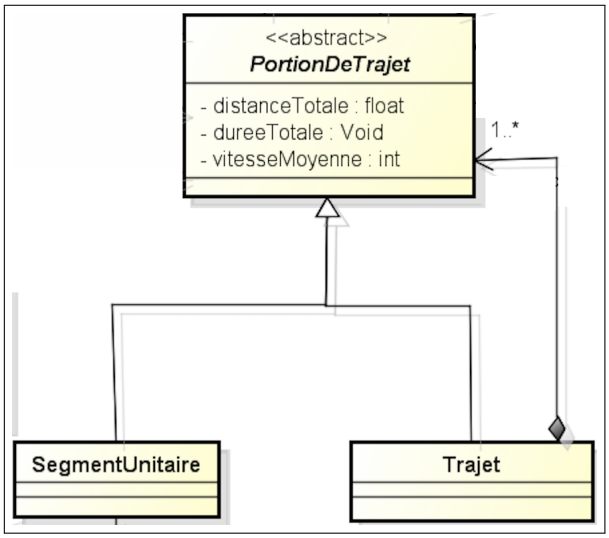
Après avoir modélisé la composante unitaire d'une trace GPS, c'est-à-dire le POI, il faut regarder de plus près dans quoi il s'inscrit. Au niveau GPX, cela correspondrait au *segment*. Mais là encore, la sémantique qui se cache derrière le segment GPX est trop figée et réductrice. Pour rappel, un segment GPX correspond à une suite de points continue dans le temps tant qu'il n'y a pas d'interruption du signal.

Mais il ne faut pas oublier que l'on souhaite étudier les déplacements d'individus et que à ce titre, on préfèrera la notion de **trajet**. En effet, le trajet d'un individu pour aller d'un point à un autre peut être parsemé d'interruptions de signal GPS pour un certain nombre de raisons et il y aura donc présence dans la trace de plusieurs segments. Pour autant, au niveau fonctionnel de notre vision, cela ne constituera qu'un seul trajet.

Dans le cadre des traitements qui seront appliqués sur ces trajets, il a même paru nécessaire

de descendre un cran plus bas dans le niveau de granularité en parlant de **segment unitaire** d'un trajet. Un segment unitaire est donc la plus petite unité de trajet. Il s'agit d'une **portion de trajet** constituée de seulement deux points (POI) entre lesquels il n'y a aucune information connue.

La modélisation de ces notions a pu être réalisée grâce à l'utilisation du *design pattern* ***"Composite"***, comme illustré dans l'exemple de la figure ci-contre.



On obtient ainsi la classe **Trajet** qui est composée de **PortionDeTrajet**. Un **SegmentUnitaire** étant une **PortionDeTrajet**.

Ce découpage du pattern *Composite* permet de traiter une **PortionDeTrajet** indépendamment du fait qu'il s'agisse d'un **Trajet** ou bien d'un **SegmentUnitaire**.

Classes de traitement

Parmi les traitements à réaliser, notamment ceux mentionnés dans la section 2.2, il a fallu réfléchir à la manière de les modéliser. Quels que soient les traitements effectués, ce qu'ils prennent en entrée et ce qu'ils retournent est toujours identique. Il s'agit d'une portion de trajet.

Ainsi il y a donc des algorithmes de traitements qui prennent une portion de trajet en entrée et renvoient une portion de trajet modifiée en sortie. Il a été choisi de qualifier ces algorithmes de **Filtre** puisqu'ils filtrent tous des points dans une portion de trajet, selon différents critères.

Le cas de figure enoncé ci-dessus nous permet d'en arriver à la justification de l'utilisation du *design pattern* **Strategie**. En effet, ce dernier est parfaitement adapté à la situation. La modélisation résultante inclut donc une interface **IFiltre** qui expose une méthode *Traiter()* renvoyant une portion de trajet.

La classe **PortionDeTrajet** pourra donc utiliser cette méthode de traitement via un objet de type *IFiltre* derrière lequel pourra sa cacher n'importe quel algorithme de traitement de son choix du moment que ce dernier implémente l'interface *IFiltre*.

3.5 Méthodes principales de traitement

3.5.1 Réduction du nombre de points

L'algorithme de réduction des traces GPS est la première étape dans le processus de traitement d'une trace GPS. En effet, une telle trace peut être composée de plusieurs dizaines de milliers de points, ce qui peut considérablement allonger les temps de traitements.

L'enjeu de la réduction du nombre de points est ainsi de pouvoir supprimer un certain nombre de points d'une trace GPS en perdant le minimum d'informations, c'est-à-dire en conservant au maximum la cohérence du tracé.

La littérature sur le sujet a rapidement montré qu'il existait un algorithme de simplification de polyligne allant dans ce sens, l'algorithme de **Douglas-Peucker**. On peut se représenter la simplification d'une polyligne par l'illustration suivante :

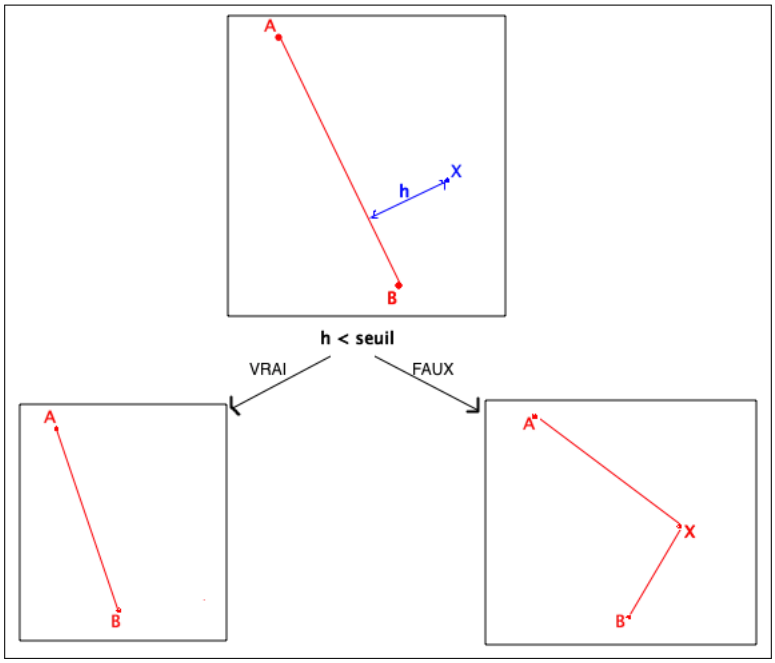
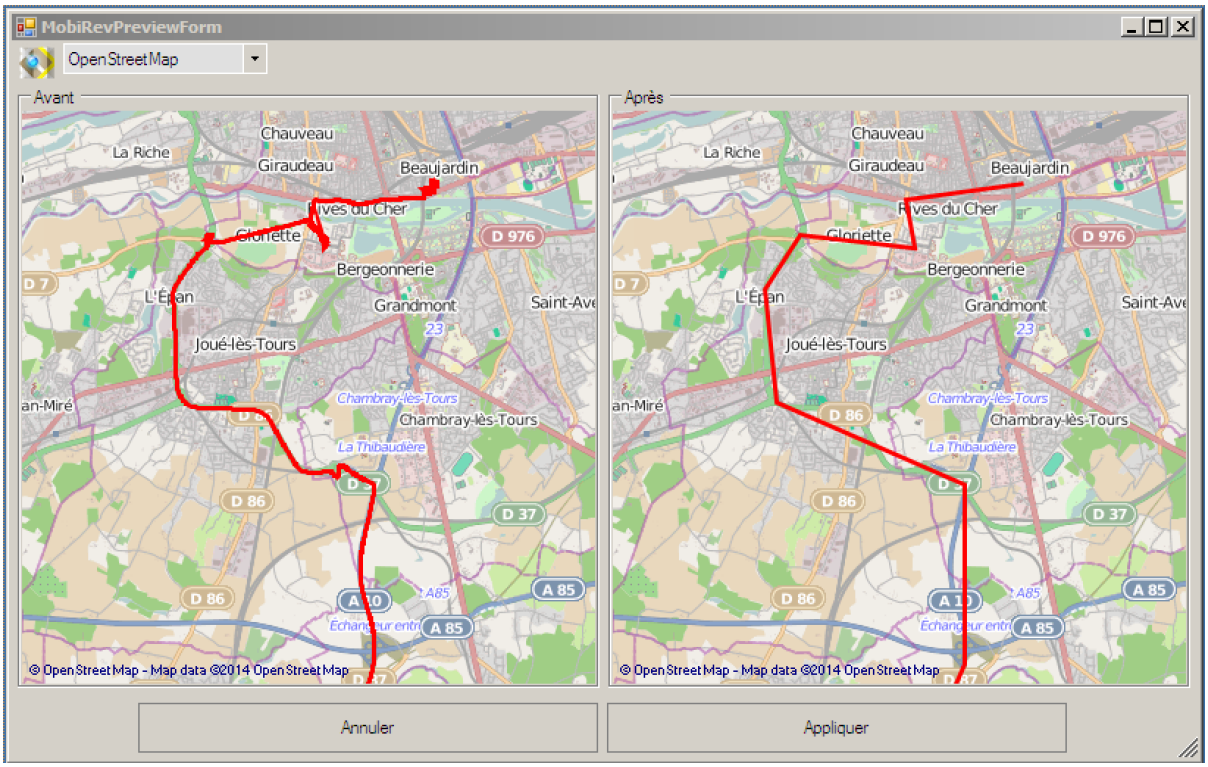


Illustration de la simplification d'une polyligne basée sur un seuil.

C'est donc l'algorithme de simplification de polyligne de Douglas-Peucker qui est implémenté et utilisé dans l'application MobiRev finale. Son fonctionnement peut être décrit selon le pseudo-algorithme suivant :

1. On prend le premier et le dernier point du trajet (nommés A et B)
2. On cherche le point (X) le plus éloigné de la ligne formée par AB
3. Si la distance orthogonale de X au segment AB est supérieure à un certain seuil, on garde X et on relance récursivement en prenant A,X puis X,B
4. Sinon on recommence au 2.
5. L'algorithme s'arrête lorsque tous les points ont été traités

Sur l'image ci-dessous, un exemple d'application de l'algorithme de Douglas-Peucker présentant une perte volontaire d'informations afin d'en illustrer le fonctionnement et le résultat :



Exemple d'application de l'algorithme de Douglas-Peucker

Ainsi, après l'implémentation et les tests de cet algorithme, ce dernier s'est avéré extrêmement efficace sur toutes les traces avec lesquelles il a été testé. Le seuil de tolérance par défaut qui a été fixé dans l'appplication est de 0.00005, ce qui correspond à une distance orthogonale d'environ 5 mètres. Ce seuil est néanmoins paramétrable par l'utilisateur via l'interface graphique prévue à cet effet.

Par exemple, une trace contenant près de 19500 points a pu être réduite à seulement 3000 sans quasiment aucune perte d'informations quant aux rues et routes empruntées par l'individu porteur du dispositif GPS. Ce qui est très concluant et permet à des traitements ultérieurs de s'exécuter plus rapidement.

Complexité

Outre l'efficacité remarquable de cet algorithme, sa complexité temporelle est également un point positif. En effet, l'algorithme de Douglas-Peucker a dans le pire des cas une complexité en $O(n^2)$ et une complexité moyenne en $O(n \times \log n)$. En conséquence, pour un test sur une trace de près de 20000 points, le temps d'exécution de l'algorithme est de l'ordre de la seconde.

3.5.2 Suppression des points aberrants

La fonctionnalité de suppression des points aberrants recouvre un certain nombre de cas de figures. Cela englobe aussi bien les points ouvertement hors-norme (vitesse ou accélération inhumaine) mais aussi les points dans la norme mais jugés humainement aberrants de part la localisation (contre-sens sur la route, rouler sur le trottoir, etc.).

Dans le premier cas de figure, ces points aberrants peuvent être facilement discriminés. Il s'agit simplement de comparer des valeurs numériques à des seuils fixés ou non par l'utilisateur. Cette première passe permet de purifier légèrement une trace donnée.

Cependant, en ce qui concerne le second cas de figure, les points aberrants ne peuvent pas être détectés de manière numérique. En effet, ni la vitesse ni l'accélération, lorsqu'elles sont cohérentes, ne permettent de savoir si des points ne sont pas parfaitement sur la route ou non. Seules les coordonnées GPS (latitude et longitude) permettent de le savoir et il faut pour cela utiliser un service qui fournisse des informations géographiques de cet ordre.

Afin de faire parfaitement "coller" les points à la route, deux méthodes hautement expérimentales ont été envisagées et mises en place.

3.5.2.1 Snap To Road

Le *Snap To Road* est une méthode qui permet, pour un point GPS donné, de le ramener sur la route, rue ou tout autre chemin pratiquable le plus proche. Cela nécessite bien entendu d'interroger une base de données SIG.

Cette méthode a pu être implémentée grâce à l'utilisation de la librairie GMap.NET *(voir sous-section 3.4.1)* qui fournit une couche d'abstraction qui permet d'accéder à l'API d'OpenStreetMap ou bien GoogleMap.

Dès lors, on peut mettre en place un simple algorithme itératif parcourant chaque point d'une trace pour savoir quel est le point correspondant sur le chemin pratiquable le plus proche.

Limites

Le *Snap To Road* présente cependant une limite simple. Le point le plus proche sur le chemin pratiquable peut ne pas correspondre au chemin réellement emprunté par l'usager. Ce scénario se produit notamment lorsque l'on se trouve dans un environnement où il y a une forte densité de chemins routiers proches les uns des autres. En centre-ville par exemple. Dans un tel cas de figure, la précision du GPS pouvant être approximative de dix mètres suffit à retourner un point rectifié qui est alors faussé.

De plus, d'un point de vue purement technique, le fait de demander un point rectifié via l'API implique une communication réseau. Pour cette raison combinée au nombre important de points par trace, le temps de traitement peut s'avérer extrêmement long. Parfois plus de cinq minutes. Ce qui n'est pas envisageable.

3.5.2.2 Demande d'itinéraire

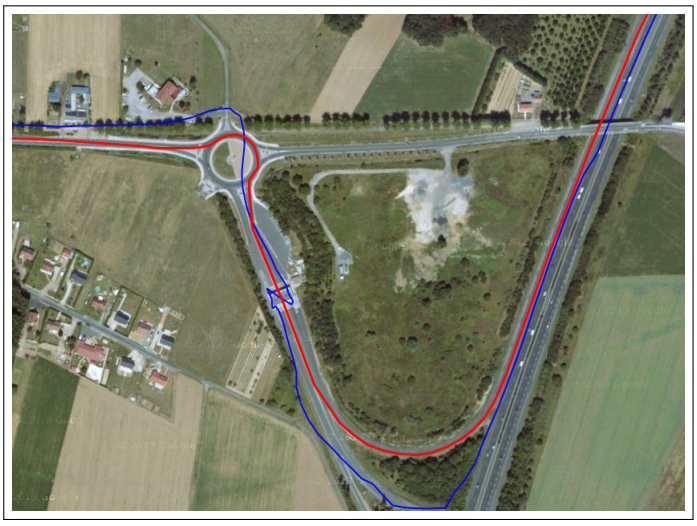
La seconde méthode envisagée pour faire "coller" les points à la route consiste également à utiliser la couche d'abstraction de GMAP.NET mais pour cette fois demander un itinéraire.

L'idée est la suivante : pour deux points GPS donnés suffisament proches entre lesquels il n'existe qu'un seul chemin possible, on demande un itinéraire entre ces deux points au service désiré (OpenStreetMap ou GoogleMap). La suite de coordonnées retournées est donc parfaitement alignée sur les chemins empruntés.

Limites

Les limites de cette méthode sont multiples. Tout d'abord, lorsqu'on demande un itinéraire entre deux points, le service (OpenStreetMap ou GoogleMap) va dans un premier temps effectuer un *Snap To Road* sur ces deux points pour avoir des points de départ et d'arrivée connus entre lesquels il peut calculer un itinéraire. Les inconvénients sont donc les mêmes que la méthode décrite précédemment.

De plus, la distance entre les points de départ et d'arrivée doit être suffisament petite pour qu'il n'existe qu'un seul chemin possible entre les deux afin de ne pas dénaturer la trace. Or, cette distance peut grandement varier sur une trace complète d'une journée. Ce qui rend un simple algorithme itératif peu efficace. À cela s'ajoute le temps d'exécution pour la même raison que la méthode *snap-to-road*.

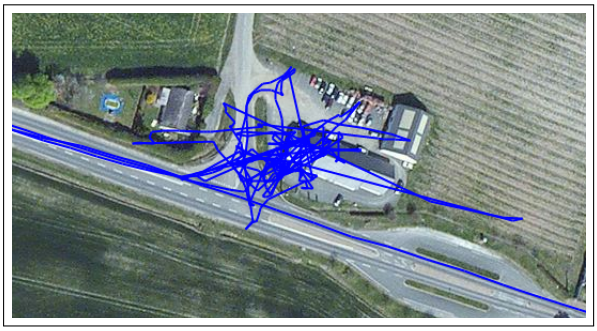


Les deux méthodes décrites ci-dessus (snap-to-road et demande d'itinéraire), bien qu'expérimentales et présentant des limites, montrent parfois des résultats concluant sur des courtes portions de trajet à rectifier qui sont sélectionnées manuellement par l'utilisateur. C'est pourquoi cette possibilité a été ajoutée à l'application.

Par ailleurs, ces méthodes sont perfectibles et demeurent néanmoins une piste à explorer de par leur potentiel. Elles pourront ainsi faire l'objet de futurs projets au sein de Polytech Tours.

Sur l'image ci-dessus, un exemple de test d'une rectification concluante sur une courte portion de trajet avec, en bleu, le trajet original et en rouge le trajet rectifié.

3.5.3 Traitement des points d'arrêts



Le traitement des points d'arrêts constitue une tâche non triviale et nécessitant de la réflexion. Pour rappel, on appelle "point d'arrêt" la totalité des points accumulés dans une zone dû à l'arrêt temporaire de l'individu porteur du dispositif GPS, ce qui crée ainsi une "pelotte" de points non discernables et plus ou moins conséquente.

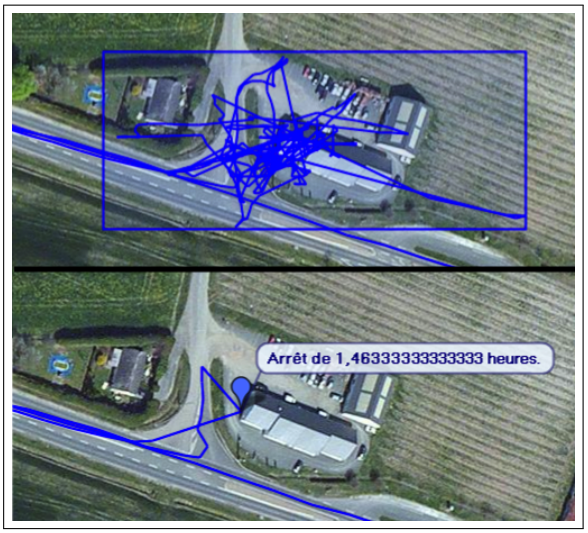
Du fait de la grande hétérogénéité des points présents dans ce type de pelotte en termes de coordonnées, il n'est pas chose aisée de les discriminer en se basant sur des informations telles que la vitesse ou bien l'accélération. Cette problématique a notamment donné lieu à plusieurs projets d'algorithmique de $3^{ème}$ année.

3.5.3.1 Sélection manuelle

Dans d'autres fonctionnalités annexes de l'application, il était nécessaire de connaître les différents points d'arrêt présents dans une trace. Typiquement, lors de l'élaboration de l'agenda qui doit en tenir compte.

Pour se faire, et dans un premier temps, il a donc été mis en place une fonctionnalité de **sélection manuelle** par l'utilisateur sur la carte pour que ce dernier puisse indiquer facilement où se trouvent les pelottes afin d'y appliquer une méthode barycentre. Comme l'illustre l'impression écran ci-contre.

Cette alternative permet ainsi de fournir à l'utilisateur final un moyen simple et ergonomique pour traiter le cas de figure des points d'arrêts sans y passer trop de temps. Cela répond au besoin.



Cette première fonctionnalité a donc pu me servir de point de départ pour obtenir des points d'arrêts sur lesquels travailler. Cependant, le but final recherché pour l'utilisateur était d'avoir une méthode de traitement qui traiterait l'intégralité d'une trace automatiquement pour en ressortir les points d'arrêts.

3.5.3.2 Détection automatique : Littérature

Afin de mettre en place un algorithme de détection automatique des points d'arrêt, il a d'abord fallu effectuer quelques recherches sur le sujet. Bien que la littérature sur ce point précis ne soit pas des plus abondantes, on trouve néanmoins des ressources.

Methodology for Converting GPS Navigational Streams to the Travel-Diary Data Format [7]

Un premier article issu de l'université du Texas traite du fait de passer de données GPS brutes à une sorte d'agenda des déplacements. Il traite pour cela de la détection des points d'arrêt. Dans cet article, aucun algorithme n'est détaillé spécifiquement mais simplement des directions sont données quant aux critères qui pourraient servir à identifier les points d'arrêt.

Le critère dont il est question dans l'article est mentionné en tant que *dwell-time threshold* qui pourrait se traduire par la limite de temps d'arrêt. Derrière cette notion intuitive se cache le fait de vérifier la durée pendant laquelle le véhicule porteur du dispositif GPS n'est pas en mouvement pour en déduire un point d'arrêt.

Si l'idée est bonne, le problème est que dans le contexte de cet article, le dispositif GPS dont il est question offre la possibilité de reconnaître les périodes de "non-mouvement", ce qui n'est pas notre cas ici.

Transportation Activity Analysis Using Smartphones [8]

Un autre article scientifique, issu du MIT (*Massachusetts Institute of Technology*), aborde la question de la détection des points d'arrêt de manière un peu plus concrète. On y trouve en effet un algorithme qui semble efficace et qui utilise trois critères pour fonctionner :

- La distance parcourue entre deux points dans une fenêtre de temps. Deux positions successives se trouvant dans une fenêtre de temps supérieure à T secondes et distante de moins de D mètres seront considérées comme appartenant à un même point d'arrêt.

- Le changement dans la précision du mode de détection de la position. Dans cette étude, les localisations sont récupérées à la fois via GPS et par le réseau cellulaire GSM. Si un individu se trouve en intérieur par exemple, le GPS sera beaucoup moins précis, voire perdra le signal, ce qui ne sera pas ou peu le cas par le réseau GSM.

- Le mode de transport utilisé. Cet article suppose l'utilisation, en amont, d'un algorithme plus évolué de détection des modes de transport. Une fois le moyen de transport connu, le fait de l'associer aux deux critères ci-dessus peut aider à discriminer un point d'arrêt dans certains cas.

De plus, avant d'appliquer cet algorithme de détection, les données GPS brutes font l'objet d'un

7. http://www.ce.utexas.edu/prof/bhat/ABSTRACTS/Srinivasan_Bricka_Bhat.doc

8. http://web.mit.edu/czegras/www/ccnc_demo.pdf

pré-traitement pour détecter les modes de transport utilisés, affiner les positions récoltées par le réseau GSM via des serveurs de localisation et remplacer les pertes de signals GPS par les données de localisation GSM. Pour toutes ces raisons, et même si l'algorithme paraît efficace, il n'est pas possible de l'appliquer à notre cas.

Increasing the accuracy of trip rate information from passive multi-day GPS travel datasets : Automatic trip end identification issues[9]

Cet article issu du site *ScienceDirect* traite également de la détection des points d'arrêt mais n'apporte pas d'algorithme. Comme déjà mentionné dans le premier article plus haut, on y retrouve la notion de "*dwell-time*", ou temps de séjour, afin de discriminer les points d'arrêt.

On y trouve également l'introduction d'une nouvelle notion qui est celle de "*heading change*". Cette dernière correspond au changement de direction observé, en degrés, d'un point à l'autre après un temps d'arrêt incertain pour confirmer qu'il s'agissait bien d'un point d'arrêt.

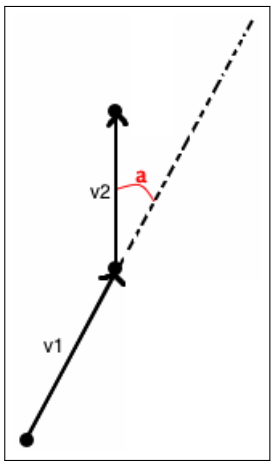
Le *dwell-time* ne peut pas servir dans notre cas car les données GPS dont il est question dans le cadre du projet MobiRev ont été enregistrées en continu même pendant des arrêts. L'intervalle entre deux enregistrements est donc toujours compris entre une et trois secondes ce qui ne nous permet pas de nous en servir comme discriminant.

En revanche, le fait d'observer le changement de direction entre les points semble pertinent dans notre cas car l'enregistrement en continu de la position crée des amats de points dont la direction de l'un à l'autre peut varier drastiquement.

9. http://www.sciencedirect.com/science/article/pii/S0965856406000425

3.5.3.3 Algorithme de détection automatique

L'algorithme de détection automatique qui a été implémenté pour répondre au besoin est basé sur un critère simple qui se trouve être l'angle entre les différents vecteurs vitesse successifs qui composent une trace. En effet, dû à l'enregistrement en continu des données GPS par le dispositif ainsi que le faible degré de précision en intérieur, les temps d'arrêt engendrent des pelottes de points dans lesquelles il y a beaucoup de croisements.



Le principe de l'algorithme est donc le suivant : étant donné un vecteur vitesse, l'algorithme va observer les vecteurs vitesse suivants par rapport aux précédents en mesurant à chaque fois l'angle formé par le vecteur $\vec{v2}$ et la droite lancée par le vecteur $\vec{v1}$. Si cet angle est supérieur à un certain seuil, 90 degrés par exemple, et que le phénomène se répète consécutivement assez souvent, on peut en déduire que l'on se trouve dans une pelotte qui caractérise un point d'arrêt.

L'algorithme en pseudo-code qui va suivre est celui utilisé dans l'application. Il s'arrête dès qu'il trouve un point d'arrêt ou bien renvoie la portion de trajet initiale le cas échéant :

Algorithm 1 Détection automatique des points d'arrêts

```
1: POUR CHAQUE (point p dans la portion de trajet) FAIRE
2:  SI (angle entre [p-1,p] et [p,p+1] > D degrés) ALORS
3:   Ajout à la liste des points d'arrêts potentiels
4:  SINON SI (nombre de fois où l'angle est plus petit que D degrés < tolérance T) ALORS
5:   Ajout à la liste des points d'arrêts potentiels
6:   Incrémentation du nombre de fois où l'angle est plus petit que D degrés
7:  SINON
8:   SI (nombre de points d'arrêts potentiels > N) ALORS
9:    On considère avoir trouvé un point d'arrêt
10:   On remplace les points d'arrêts potentiels par le barycentre et on calcule le temps d'arrêt
11:   RETOURNER La portion de trajet modifiée
12:  FINSI
13: FINSI
14: FINPOUR
15: RETOURNER La portion de trajet initiale          ▷ Aucun point d'arrêt trouvé
```

Les variables D, T et N sont les données à modifier afin de calibrer au mieux l'algorithme de détection automatique. Une tolérance T trop faible pourra entrainer la suppression de petits points d'arrêt ou bien en segmenter un plus gros. Une valeur N trop faible pourra créer des *faux positifs* tandis qu'une valeur trop élevée pourrait faire passer sous silence des points d'arrêt potentiels.

Ces variables ont été affectées de valeurs par défaut suite à de nombreux tests sur plusieurs traces

afin d'offrir un bon compromis quant au taux d'erreur de détection. Ces valeurs sont les suivantes : $D = 70$, $T = 10$ et $N = 50$.

Comme mentionné plus haut, cet algorithme de détection automatique s'arrête dès qu'il trouve un point d'arrêt. Il est donc utilisé dans une boucle de plus haut niveau qui donne simplement le pseudo code suivant :

Algorithm 2 Boucle de détection automatique des points d'arrêts

```
ENTRÉE : portionDeTrajet

FAIRE
  portionDeTrajet ← DetectionAutomatique(portionDeTrajet)
TANT QUE (portionDeTrajet est modifiée)

RETOURNER portionDeTrajet
```

Cet algorithme basé sur l'angle entre les vecteurs vitesse fournit donc un moyen rapide et assez efficace pour obtenir un premier agenda comportant potentiellement des erreurs mais qui permet à l'utilisateur d'avoir une rapide introduction dans la compréhension globale de la trace. C'est ce que désirait la maîtrise d'ouvrage.

Complexité

Nous allons voir que la complexité de l'algorithme de détection automatique des points d'arrêt peut être approximée par $O(n)$. Définissons quelques variables :

— Soit n : le nombre de POI dans la portion de trajet
— La signification des variables D, T et N est la même que celle mentionnée plus haut. On s'intéresse ici à la variable N qui représente le nombre minimum de POI pour constituer un point d'arrêt.

Voyons maintenant les différents cas de figure possibles et leur complexité dans le pire des cas :

- L'algorithme ne trouve aucun point d'arrêt :

  La boucle *POUR CHAQUE* de plus haut niveau va parcourir les n POI dans la portion de trajet avant de retourner la portion de trajet initiale. La boucle *TANT QUE* qui appelle la détection automatique se stoppera à la première itération. La complexité sera donc en ***O(n)***.

- La portion de trajet toute entière constitue un point d'arrêt :

  La boucle *POUR CHAQUE* de plus haut niveau va parcourir les n POI dans la portion de trajet avant de retourner une portion de trajet modifiée. La boucle *TANT QUE* qui appelle la détection automatique itérera une seconde fois. Lors du second appel à la détection automatique, tous les précédents POI ont été fusionnés en un point d'arrêt. La portion de trajet initiale sera immédiatement retournée après une seule itération.

La complexité peut donc être ici aussi approximée par ***O(n)***.

- La portion de trajet ne contient qu'un seul point d'arrêt :

  Dans le pire des cas, la boucle *POUR CHAQUE* de plus haut niveau détectera l'unique point d'arrêt sur les N derniers POI parmi les n de la portion de trajet. Il y aura donc un premier parcours des n POI. Ces N derniers POI seront fusionnés avant que la boucle *TANT QUE* ne rappelle la détection automatique qui cette fois parcourra les $n - N$ POI restants avant de retourner une portion de trajet non modifiée.
  La complexité dans ce cas serait en $O(n + (n - N))$ ce qui peut s'approximer par ***O(n)***.

- La portion de trajet contient $\frac{n}{N}$ points d'arrêt, le nombre maximum détectable :

  Dans ce scénario, l'algorithme de détection automatique découvrirait un point d'arrêt tous les N POI.

  Itération 1 : La boucle *POUR CHAQUE* retournerait une portion de trajet modifiée au bout de N POI.
  Itération 2 : La boucle *POUR CHAQUE* retournerait une portion de trajet modifiée au bout de $N + 1$ POI. I.e. le premier étant le résultat de la fusion de l'itération précédente.
  Itération 3 : La boucle *POUR CHAQUE* retournerait une portion de trajet modifiée au bout de $N + 2$ POI.
  Itération n : La boucle *POUR CHAQUE* retournerait une portion de trajet modifiée au bout de $N + (\frac{n}{N} - 1)$ POI.

  Finalement, la boucle *TANT QUE* va itérer $\frac{n}{N}$ fois durant lesquelles le nombre d'itérations sera égal à : $N + (nombre\_de\_points\_d'arrêt\_précédemment\_détectés)$. La complexité de ce scénario étant donc de $O(\frac{n}{N} \times N + \frac{n}{N}) = O(n + \frac{n}{N})$.

  La complexité résultante peut donc de nouveau être approximée par ***O(n)***.

3.5.3.4 Amélioration de la détection automatique

À la vue des résultats donnés après l'application de l'algorithme vue précédemment, un algorithme de post-traitement a pu être envisagé dans le but de corriger certaines erreurs de détection.

En ce qui concerne les points d'arrêt qui auraient été perdus à cause d'une valeur de tolérance T trop faible ou bien d'une valeur de N trop élevée, rien ne peut être fait. En revanche, pour les longs points d'arrêt qui auraient été détectés de manière segmentée, un algorithme a pu être mis en place afin de corriger ceci.

Cet algorithme de post-traitement va donc itérer sur les points d'arrêt détectés et vérifier pour chacun d'entre eux les points d'arrêt environnants afin de voir s'il est possible de les fusionner en fonction du temps et de la distance qui les séparent. Le pseudo-code qui va suivre est l'algorithme qui s'arrête dès qu'il trouve des points d'arrêt à fusionner. Il renvoie la trace modifiée ou bien la trace initiale le cas échéant :

Algorithm 3 Amélioration de la détection automatique des points d'arrêts

ENTRÉE 1 : tailleZone ▷ rayon de la zone de recherche autour d'un point d'arrêt
ENTRÉE 2 : intervalle ▷ intervalle maximal de temps entre deux points d'arrêt

POUR CHAQUE (point d'arrêt p dans la portion de trajet) **FAIRE**
  Ajouter p à la liste des points d'arrêt à fusionner
  POUR CHAQUE (point d'arrêt à moins de *tailleZone* autour de p **ET** séparé de moins de *intervalle*) **FAIRE**
    Ajout à la liste des points d'arrêt à fusionner
  FINPOUR
  SI (nombre de points d'arrêt à fusionner > 1) **FAIRE**
    Fusionner les points d'arrêt dans la liste des points d'arrêt à fusionner
    RETOURNER La portion de trajet modifiée
  FINSI
  Reset de la liste des points d'arrêt à fusionner
FINPOUR
RETOURNER La portion de trajet initiale ▷ Aucun point d'arrêt trouvé

Comme précédemment, cet algorithme s'arrête dès qu'il trouve des points d'arrêt à fusionner. Il est donc encapsulé dans une boucle de plus haut niveau qui l'appelle :

Algorithm 4 Boucle appelant le post-traitement sur les points d'arrêt

ENTRÉE : portionDeTrajetAvecPointArrêt

FAIRE
  portionDeTrajet ← AmélioriationPostTraitement(portionDeTrajetAvecPointArrêt)
TANT QUE (portionDeTrajet est modifiée)
RETOURNER portionDeTrajet

Les paramètres $tailleZone$ et $intervalle$ sont paramétrables par l'utilisateur via l'interface prévue à cet effet. Ces derniers sont respectivement pré-réglés sur deux kilomètres et une minute.

Complexité

Pour un nombre n de points d'arrêt présents dans une portion de trajet donnée en entrée, la complexité de cet algorithme de post-traitement peut être approximée par ***O(n)***.

Cependant, dans la pratique, sur toutes les traces testées et en envisageant le pire des scénarios, le nombre de points d'arrêt dépasse rarement la vingtaine. C'est pourquoi cet algorithme de post-traitement s'exécute rapidement.

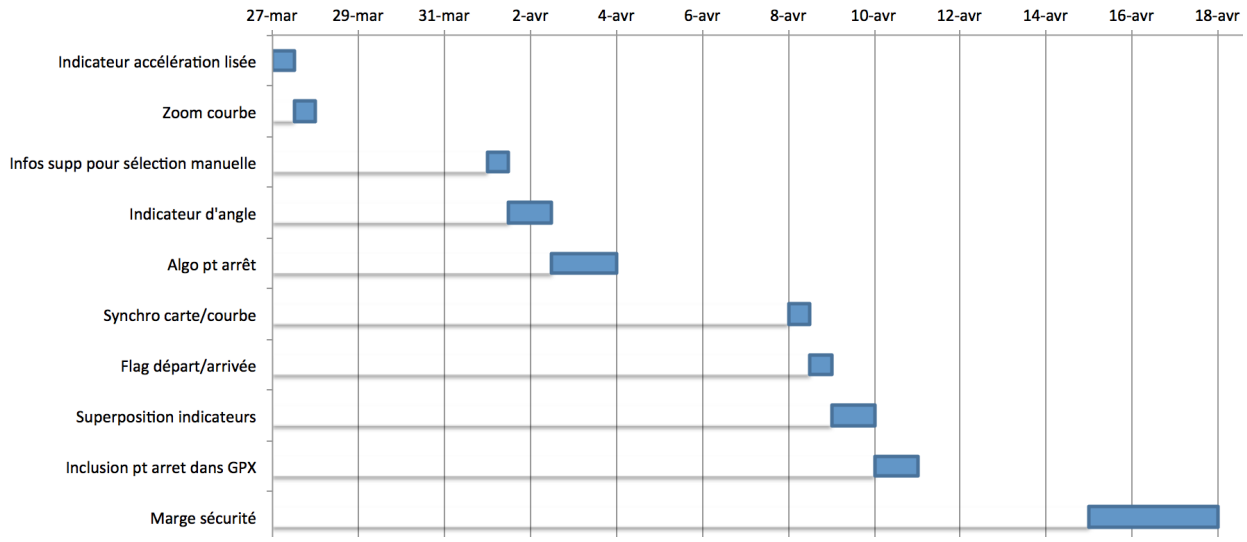
Phase de développement

4.1 Planning de développement

Après toute la phase d'étude et de conception, c'est au mois de janvier que les développements allaient devoir commencer. J'ai donc dans un premier temps tenté d'élaborer une liste la plus exhaustive possible des tâches restantes à accomplir jusqu'à la fin du PFE en y précisant les échéances.

Comme mentionné dans la sous-section 2.3.1.1, dans le cadre de la démarche AGILE adoptée pour le projet, des *sprints* étaient définis au début de chaque mois avec Carl ESSWEIN afin de déterminer les tâches à venir et les échéances pour le mois en cours.

Voici en exemple le planning du dernier mois (fin Mars/Avril) du PFE :



Exemple d'un planning de développement

J'avais donc un planning de développement similaire par mois dont les deadlines, jusqu'a la fin du projet, ont toujours pu être respectées. Ceci a permis, à mon encadrant comme à moi-même, de vérifier la bonne marche du projet en s'assurant qu'aucun retard n'était pris.

Il est à noter que la rédaction du présent rapport ainsi que de la documentation était faite en parallèle de ces plannings.

4.2 Démarche

4.2.1 Architecture

L'architecture logicielle qui a été choisie pour l'application MobiRev a été l'architecture *MVC*[1] et ceci pour plusieurs raisons.

Il s'agit tout d'abord de l'architecture privilégiée lorsque les développements se font avec le framework .NET pour l'élaboration d'une application lourde pour Windows. En effet, la segmentation des classes, les composants graphiques et les évènements vont naturellement dans ce sens.

Ainsi le diagramme des classes métier, dont il a déjà été question dans la sous-section 3.4.2, recense les classes qui rentreront dans la catégorie ***M**odèle* du MVC. La ***V**ue* et le ***C**ontrôleur* étant les autres classes techniques d'une application .NET (composants graphique, formulaire, etc.).

Par ailleurs, le modèle *MVC* favorise grandement la modularité et donc par la même occasion la maintenabilité de l'application. C'est évidemment un point important puisque cette application va très certainement être amenée à être reprise dans le futur en vue d'y apporter des améliorations.

En plus de la documentation technique, il était donc important de tout mettre en oeuvre pour faciliter la transition lors d'une probable reprise de cette application, ce qui passe par le choix d'une architecture adaptée.

4.2.2 Tests

4.2.2.1 Tests boîte noire

Le test dit *boîte noire* est généralement le premier qui est effectué. De manière basique, il correspond simplement au lancement et à l'utilisation du logiciel, tel un utilisateur quelconque.

Ce premier test permet d'une part de s'assurer que l'interface gaphique est ergonomique et conforme à celle attendue. Il permet de surcroît de vérifier la cinématique et le bon fonctionnement des diverses fonctionnalités implémentées.

La manipulation en boîte noire du logiciel par moi-même ainsi que mon encadrant a permis, à plusieurs reprises, de faire emmerger des cas de figures causant des dysfonctionnements de l'application.

4.2.2.2 Tests unitaires

Le test unitaire correspond au fait de tester une méthode de traitement en particulier. Il s'agit de donner en entrée une série de données qui doit engendrer une issue connue à l'avance (échec ou réussite) et de vérifier que la méthode répond correctement.

1. **M**odèle **V**ue **C**ontrolleur

C'est ainsi que j'ai procédé notamment pour les trois méthodes de traitement décrites plus haut dans ce rapport. Pour des raisons évidentes, des tests unitaires n'ont pas été réalisés mécaniquement sur toutes les méthodes de l'application mais uniquement sur les plus complexes. Les autres, assez courtes et/ou triviales, n'en avaient pas le besoin.

4.2.2.3 Tests de regression

Les tests de regression permettent de s'assurer que des fonctionnalités ne cessent pas de fonctionner après une modification qui les concernent ou non.

Ainsi, après chaque nouvelle implémentation de fonction, l'application a été testée avec les autres tests vus précédemment afin de s'assurer que le programme n'était victime d'aucune régression de fonctionnalité.

4.2.2.4 Tests de performance

Dans le cadre de l'application MobiRev, il n'a jamais été question de critère de performance, tant sur le plan mémoire que sur le temps d'exécution. Cependant, dans les cas de figures comportant des traitements lourds et multiples sur des traces GPS, les temps d'exécution avaient parfois tendance à être assez élevés et pouvant donc entraver l'expérience utilisateur.

J'ai donc utilisé un logiciel de *profiling* (celui intégré dans *Visual Studio*) pour me permettre de voir et comprendre dans quel traitement précisément le logiciel passait son temps d'exécution. Cette démarche m'a permis de paralléliser, quand c'était possible, certains points charnières dans le code pour en accélérer la vitesse.

L'utilisation d'une programmation multi-threads a également permis de conserver une interface graphique fluide pour une expérience utilisateur la moins troublée possible lors de traitements lourds.

Recettage

5.1 Recettes intermédiaires

Dans tout projet de développement logiciel, la phase de recettage auprès du client est une étape importante. Ce projet ne fait pas exception à la règle.

Cependant, puisque l'on se trouve dans le cadre d'une démarche projet AGILE, il n'y a pas eu qu'une seule recette à l'issue du projet mais plusieurs recettes au fur et à mesure de l'avancement du projet.

Ainsi, dès le mois de février, une fois que les développements étaient assez avancés, des points ont pu être organisés avec Benoît FEILDEL afin qu'il puisse apprécier l'évolution de l'application, la tester et indiquer les directions à adopter notamment en termes de désir au niveau de l'ergonomie de l'interface utilisateur.

Ces points avec la maîtrise d'ouvrage ont donc fait l'objet de recettes intermédiaires permettant de s'assurer de l'état du logiciel à un instant donné et d'y apporter d'éventuelles améliorations en vu de continuer en partant d'une base valide.

Ce sont ainsi quatre recettes intermédiaires qui ont pu être organisées lors de la phase de développement, du mois de janvier au mois d'avril. Chacune de ces recettes se déroulait en commençant par un point sur les modifications apportées depuis la dernière recette. Puis une démonstration était faite au client avant que la main sur l'application ne lui soit donnée pour qu'il teste par lui-même.

5.2 Recette finale

Un peu avant l'échéance du projet de fin d'étude, les deux dernières semaines du mois d'avril, une première version finale de l'application MobiRev a été compilée et livrée au client, accompagnée d'une documentation utilisateur.

Ceci a été fait dans le but de constituer une ultime phase de recette pendant laquelle la maîtrise d'ouvrage peut tester l'application dans sa version finale et en situation réelle. Cela permet d'obtenir des retours afin d'apporter d'éventuelles modifications mineures d'ordres ergonomique ou fonctionnelle avant la fin du projet.

Conclusion

6.1 Bilan du projet

Afin de faire le bilan de ce projet, nous pouvons tout d'abord en parler en termes d'échéance. Comme mentionné au début de ce rapport, la démarche projet adoptée pour le projet était une démarche AGILE avec la notion de *sprints*. Dans ce cadre, toutes les tâches qui étaient fixées en début de mois pour chaque sprint ont toujours été achevées avec succès dans le temps imparti.

Au niveau des livrables, tout a été mis en oeuvre afin de faciliter la reprise et la compréhension du projet MobiRev. Dans un premier temps, une documentation utilisateur présentant l'interface et l'utilisation des différentes fonctionnalités a été rédigée d'une part pour la maîtrise d'ouvrage, mais également en guise d'introduction pour quiconque voudrait reprendre l'application.

De la même manière, une documentation technique complète a été générée à partir d'un code largement commenté pour faciliter et accélérer la compréhension d'un futur repreneur. Par ailleurs le présent rapport constitue également une source d'explication quant aux algorithmes clés de l'application.

Enfin, une machine virtuelle "clés en main" a été remise à mon encadrant. Cette dernière inclut un environnement de développement pré-configuré qui permet immédiatement de récupérer la dernière version de l'application MobiRev sur le SVN de Polytech Tours dans le but de la tester ou bien d'en continuer le développement.

Si ces livrables permettent une reprise facilitée de l'application, c'est également parce-que toutes les fonctionnalités mentionnées dans le cahier des spécifications n'ont pas pu être implémentées. Le projet MobiRev est en effet un chantier de longue haleine qui n'a pas vocation à se terminer avec ma contribution. Il a donc fallu lotir les fonctionnalités afin de répondre aux besoins prioritaires exprimés par Benoît FEILDEL.

Ainsi, comme nous allons le voir dans la section suivante sur le bilan des fonctionnalités, certaines fonctions optionnelles ou bien trop complexes n'ont pas pu être réalisées pour pouvoir rentrer dans le temps qui était imparti pour le projet de fin d'étude.

6.1.1 Bilan des fonctionnalités

Lorsque le cahier des spécifications a été rédigé, il a été pensé de manière à garantir une certaine traçabilité quant aux diverses fonctionnalités souhaitées. C'est pourquoi ces dernières ont été codifiées afin de pouvoir les retrouver et les lister facilement.

Voici le bilan des fonctionnalités réalisées, à mettre en parallèle avec le cahier des spécifications (voir section 7.4) :

— T001 : Import/Export
— T002 : Enregistrement du travail
— F001-01 : Réduction de la trace
— F001-02 (optionnelle) : Rectification de la trace *(A reprendre et perfectionner)*
— F002-01 : Informations primaires de la trace (F002-011, F002-012, F002-013, F002-014, F002-015, F002-016)
— F002-021 : Détection des points d'arrêt
— F003-03 : Visualisation sur fond cartographique
— F003-04 : Représentation des points d'arrêt *(A reprendre et perfectionner)*
— F003-05 : Indicateurs visuels
— F003-05 : Visualisation Agenda

Et voici le bilan des fonctionnalités qu'il reste à faire :

— F002-022 (optionnelle) : Calcul des GES
— F002-023 (optionnelle) : Détection des intersections pour inférer sur des trajets communs
— F002-03 : Informations ternaires de la trace (ex : mode de transport)
— F003-01 : Lecture temporisée

6.2 Bilan personnel

Sur un plan plus personnel, je regretterai seulement de ne pas avoir pu approfondir davantage les fonctionnalités qui n'ont pas été réalisées. Bien que je comprenne qu'il ait fallu lotir pour répondre aux besoins principaux du client.

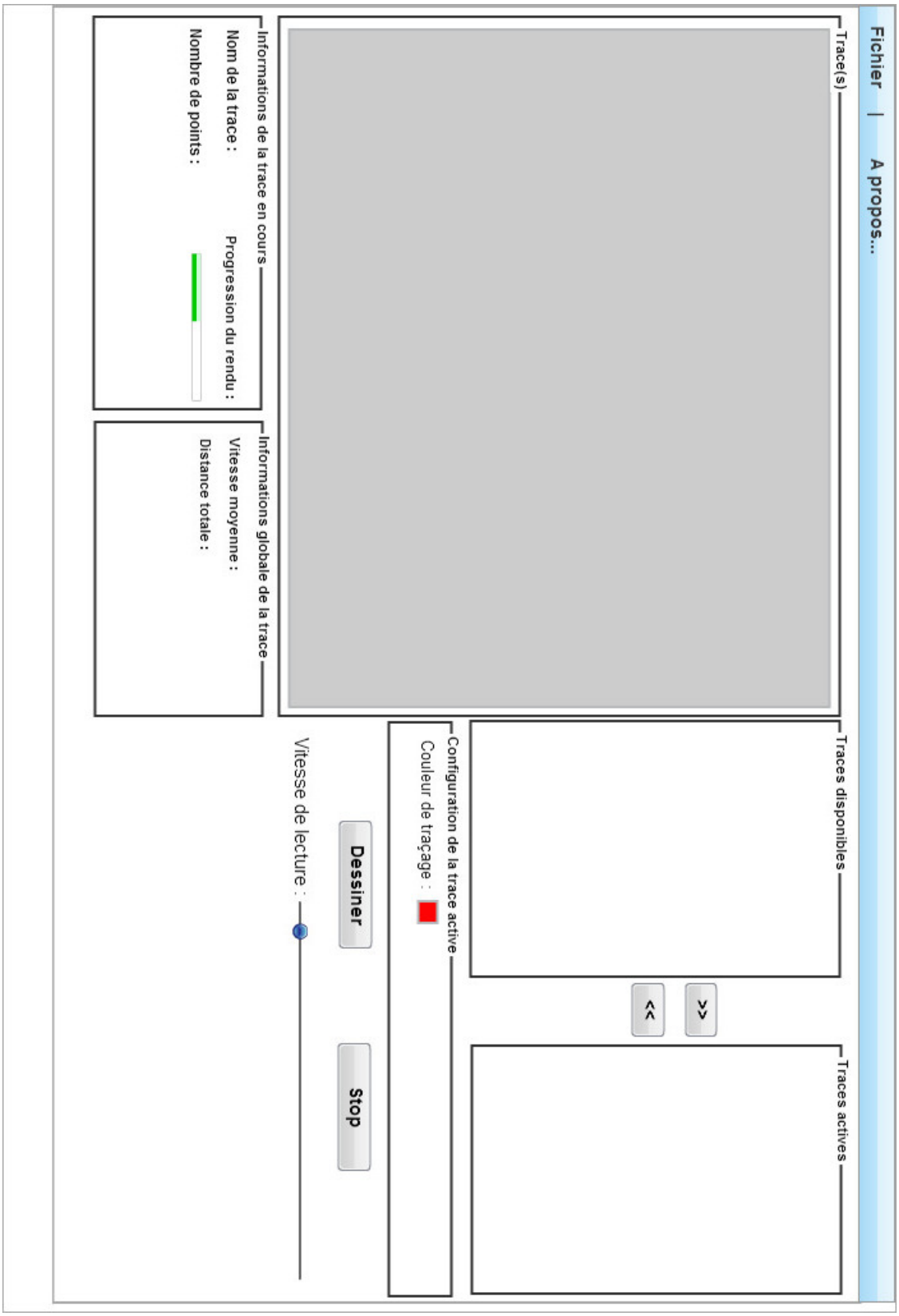
Néanmoins, j'ai globalement apprécié travailler sur ce projet. Il s'agissait en effet de mon premier choix et il s'est avéré à la hauteur de mes attentes. Tout d'abord d'un point de vue scolaire et professionnel, la mise en pratique de concepts UML sur un projet de cette échelle est très formateur. On comprend mieux l'importance d'une modélisation réussie et l'incidence que cela peut avoir sur la suite du projet. Par ailleurs la réflexion que j'ai eu à mener sur les algorithmes de traitement et leur complexité a été également très intéressante.

Mais outre l'aspect scolaire, si j'ai beaucoup apprécié ce projet c'est aussi pour son côté très "professionnalisant". Il s'agit en effet d'un projet de développement logiciel avec toutes les phases clés que cela comporte : audit du client, modélisation, réflexion algorithmique, implémentation et livraison. Cela correspond typiquement au type de travail et de réflexion qui sera attendu de nous dès le mois d'octobre prochain sur le marché du travail et c'est à mon avis un réel avantage pour moi.

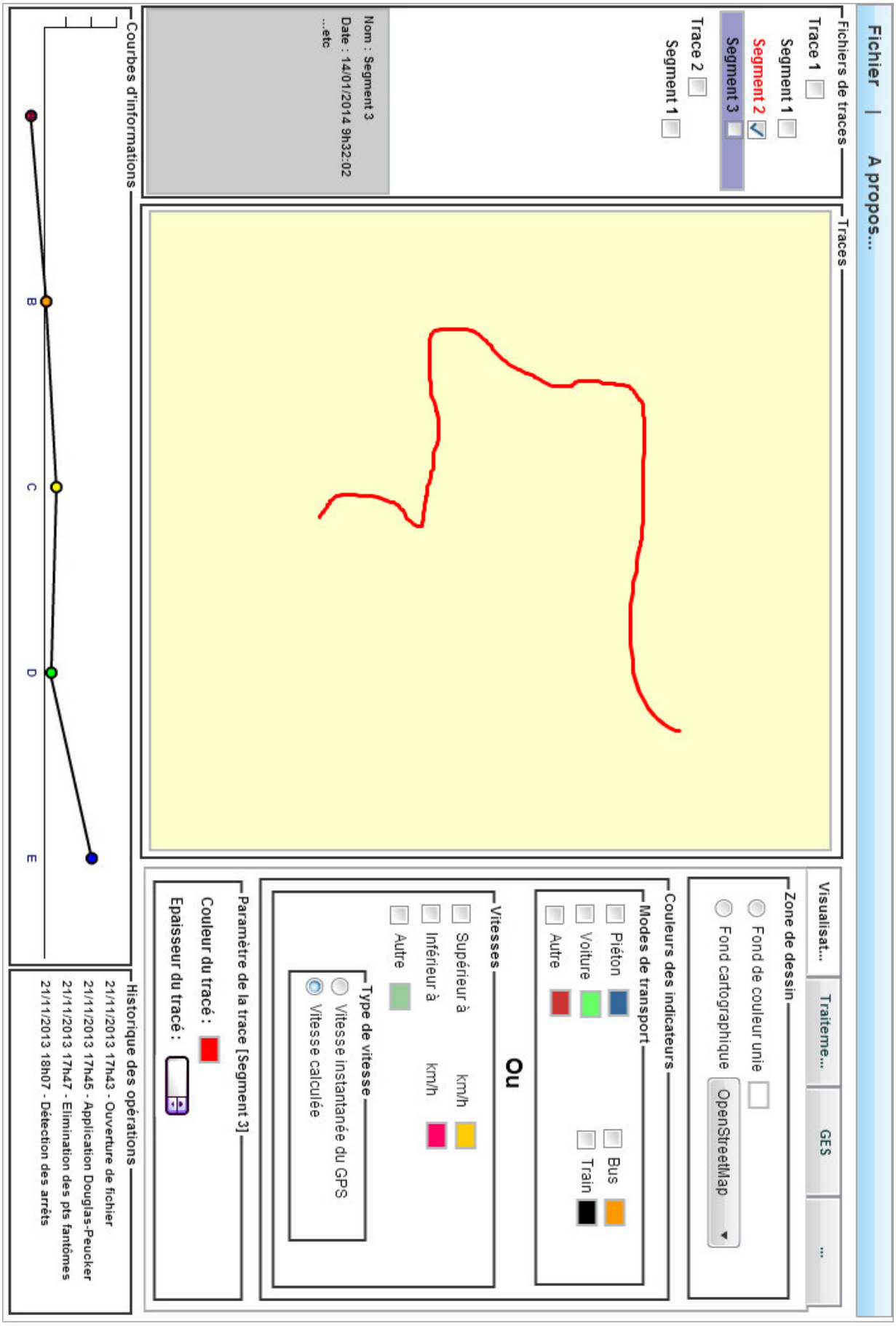
Annexes

7.1 Maquette

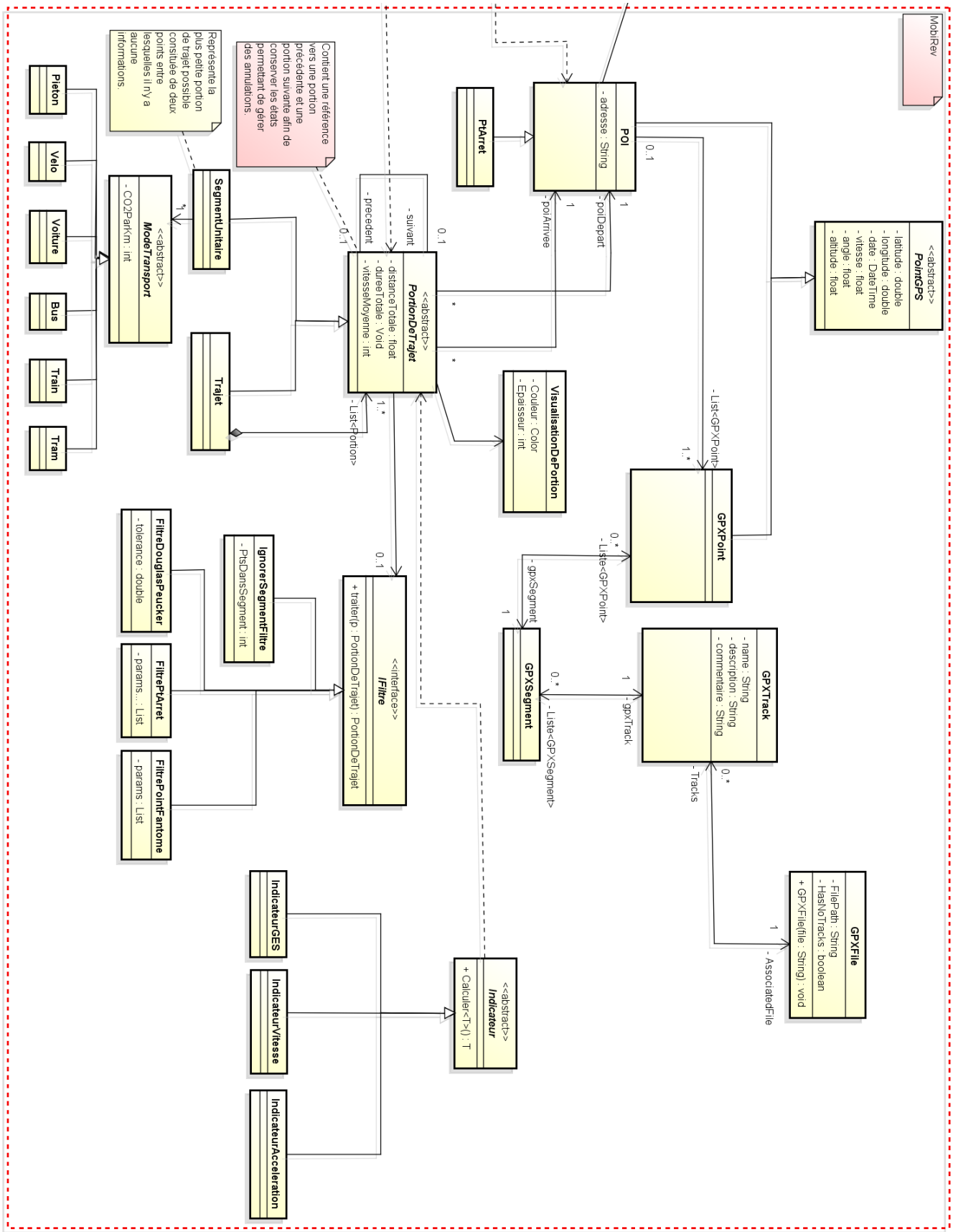
7.1.1 Version initiale



7.1.2 Version stable



7.2 Diagramme des Classes Métier



7.3 Cahier des Charges

Projet MobiRev :
Cahier des charges

Maxime SERRA
Encadré par Carl ESSWEIN

Version du 6 novembre 2013

Table des matières

I Objectif du document

Ce document a pour but de présenter le contexte, besoins et fonctionnalités souhaitées dans le cadre du projet de fin d'étude intitulé "MobiRev. La mobilité révélée par GPS : élaboration d'un outil de traitement automatique des traces GPS".

Ce cahier des charges reste à un niveau strictement fonctionnel et reprend le contexte, besoins et fonctionnalités souhaitées qui ont été recueillis lors de la première réunion du 26/09/2013 avec Benoît FEILDEL, l'interlocuteur privilégié pour le projet.

II Contexte

Ce projet émane de l'UMR CITERES dans le cadre d'un projet de recherche sur les déplacements périurbains suite à un appel d'offre du PUCA (**P**lan **U**rbanisme **C**onstruction **A**rchitecture).

L'objectif de la recherche est d'observer sur le temps les déplacements urbains et périurbains d'un certain nombre de personne afin de voir les types de déplacements effectués (voiture, piéton, bus, train, etc.) et d'en étudier les impacts notamment en terme d'émission de CO_2.

Le but étant de mesurer la différence de ces impacts entre urbain et périurbain pour confirmer ou infirmer certaines hypothèses quant a la fréquence de certain type de transport selon la distance d'habitation.

III Besoin

Il s'agit d'élaborer un outil de traitement automatisé de traces GPS permettant de révéler et de mesurer les mobilités quotidiennes d'individus localisés dans les espaces urbains et périurbains.

L'outil de traitement s'appliquera aux traces GPS récoltées notamment dans le cadre de l'enquête PériVia (2012) auprès d'une quarantaine d'individus urbains et périurbains ainsi que sur une vingtaine de personnes dans le cadre du programme MOUR.

L'outil de traitement devra intégrer différentes procédures paramétrables permettant la suppression des points aberrants (analyse des vitesses instantanées), l'identification et la mesure des points et des temps d'arrêts (méthode barycentre), le redressement si besoin de certaines portions de la trace (courbes de bézier).

Enfin, l'outil devra permettre la visualisation du tracé obtenu après traitement et la production d'un fichier nouveau trace. La production d'indicateurs analytiques, du type émissions de CO2, est également envisageable.

Cet outil de traitement devra également permettre, en analysant des traces GPS, d'inférer sur ces dernières dans le but d'en déduire des informations complémentaires telles que le mode de transport.

IV Fonctionnalités souhaitées

Les fonctionnalités souhaitées qui vont être présentées ici ne sont pas détaillées au plus bas niveau. Elles sont expliquées comme un souhait en faisant parfois abstraction de certaines contraintes et aucune solution pour les réaliser n'est donnée. Elles seront plus détaillées et découpées dans le cahier des spécifications système.

Enfin, la codification *FXXX* mentionnée pour chaque fonctionnalité servira de repérage plus tard pour faire l'analogie avec le cahier des spécifications systèmes.

A Traitement des traces GPS (F001)

Il s'agit dans un premier temps de pouvoir effectuer un post-traitement sur les traces GPS recueillis afin d'en éliminer les points incohérents dit "fantômes" pour avoir une traces GPS la plus propre et cohérente possible.

Dans un second temps, il faut également pouvoir détecter les points qui correspondent à un arrêt des déplacements.

B Analyse des traces GPS (F002)

Il s'agit de pouvoir analyser les traces GPS dans le but d'en ressortir un certain nombre d'informations telles que la vitesse de déplacement, le type de moyen de transport ou encore les émissions de CO_2.

De ces informations peut également en ressortir des statistiques journalières ou hebdomadaires.

C Visualisation des traces GPS (F003)

Il s'agit de pouvoir visualiser, dans l'outil de traitement, les traces GPS précédemment lissées sur une images satellite (de Google Earth par exemple).

Il faut également pouvoir représenter les points d'arrêt précédemment détectés par des cercles dont le diamètre serait proportionnel au temps d'arrêt.

Enfin, il faudrait pouvoir avoir des indicateurs visuels sur cette visualisation des traces dans le but de représenter les informations extraites par la fonctionnalité F002. Par des couleurs par exemple.

7.4 Cahier des Spécifications

Projet MobiRev
Cahier de spécifications

Maxime SERRA
Encadré par Carl ESSWEIN

Version du 14 janvier 2014

Table des matières

I Objectif du document

Ce document a pour but de présenter le contexte, besoins et fonctionnalités détaillées dans le cadre du projet de fin d'étude intitulé "MobiRev. La mobilité révélée par GPS : élaboration d'un outil de traitement automatique des traces GPS".

Ce cahier des spécifications systèmes reprend le contexte, besoins et fonctionnalités souhaitées qui sont énoncés dans le cahier des charges. La codification des fonctionnalités reprend celle initiée dans le cahier des charges et sera plus détaillée.

II Contexte

Ce projet émane de l'UMR CITERES dans le cadre d'un projet de recherche sur les déplacements périurbains suite à un appel d'offre du PUCA [1].

L'objectif de la recherche est d'observer, sur une période assez significative, les déplacements effectués d'un certain nombre de personnes dans les contextes spatiaux urbains, périurbains et ruraux afin de voir les types de déplacements effectués (voiture, piéton, bus, train, etc.) et d'en étudier les impacts notamment en terme d'émission de CO_2.

Le but étant de mesurer la différence de ces impacts lorsque les lieux de résidence des individus enquêtés se situent dans les différents contextes spatiaux urbains, périurbains et ruraux pour confirmer ou infirmer certaines hypothèses quant a la fréquence de certain type de transport selon la distance d'habitation.

1. **Plan Urbanisme Construction Architecture** : *Le PUCA développe des programmes de recherche incitative, des actions d'expérimentations et apporte son soutien à l'innovation et à la valorisation scientifique et technique dans les domaines de l'aménagement des territoires, de l'habitat, de la construction et de la conception architecturale et urbaine.(http ://rp.urbanisme.equipement.gouv.fr/puca/puca/presentation.htm)*

III Besoin

Il s'agit d'élaborer un outil de traitement automatisé de traces GPS permettant de révéler et de mesurer les mobilités quotidiennes d'individus localisés dans les espaces urbains et périurbains.

L'outil de traitement s'appliquera aux traces GPS récoltées notamment dans le cadre de l'enquête PériVia (2012) [2] auprès d'une quarantaine d'individus urbains et périurbains ainsi que sur une vingtaine de personnes dans le cadre rurale de l'enquête MOUR (**MO**bilité et **U**rbanisme **R**ural) [3].

L'outil de traitement devra intégrer différentes procédures paramétrables permettant la suppression des points aberrants (analyse des vitesses instantanées), l'identification et la mesure des points et des temps d'arrêts (méthode barycentre), le redressement si besoin de certaines portions de la trace.

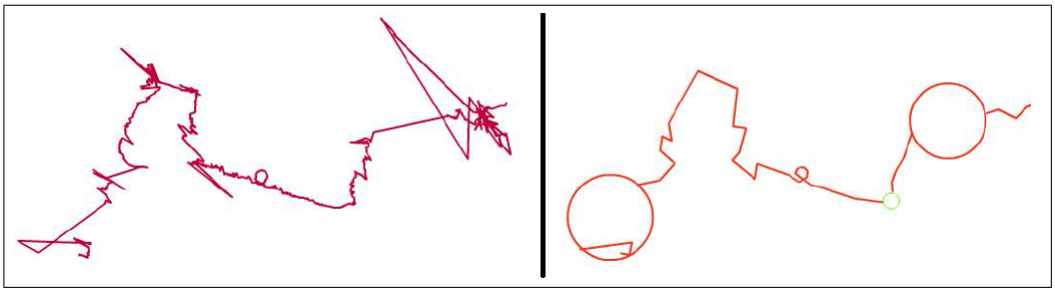


FIGURE 1 – Exemple de correction de points aberrants et détection d'arrêts

Enfin, l'outil devra permettre la visualisation du tracé obtenu après traitement et la production d'un fichier nouveau trace. La production d'indicateurs analytiques, du type émissions de CO2, est également envisagée.

Cet outil de traitement devra également permettre, en analysant des traces GPS, d'inférer sur ces dernières dans le but d'en déduire des informations complémentaires telles que le mode de transport.

2. ENQUÊTE PÉRIVIA : *http ://rp.urbanisme.equipement.gouv.fr/puca/activites/periurbain-epreuve-modeles-habiter1440.pdf*

3. ENQUÊTE MOUR : *http ://www.ort-centre.fr/Mobilite-et-Urbanisme-Rural-MOUR*

IV Spécifications technologiques

Les spécifications technologiques qui vont suivre sont le résultat de l'entretien du jeudi 10 octobre avec Benoît FEILDEL et Carl ESSWEIN.

A Type d'application

La nature de l'outil à élaborer sera une **application hors-ligne mono-poste et mono-utilisateur**. Il s'agira d'un programme exécutable classique mais devant néanmoins permettre des **connexions occasionnelles à internet**.

A moyen ou long terme, la possibilité de faire évoluer l'outil en une application web doit rester ouverte. En conséquence, cela implique que l'application soit, à la base, construite de manière extrêmement **modulaire**.

B Choix technologiques

Le choix de la technologie à employer pour réaliser l'application s'est porté sur le langage **C# avec l'utilisation du framework .NET**.

Motifs

Le choix d'utiliser la-dite technologie pour réaliser l'application a été motivé par les raisons suivantes :

- L'utilisation du framework .NET avec le langage C# est par nature très modulaire.
- Les applications logicielles construites avec le framework .NET sont aisément convertibles en une application web. Simplement, seule la partie visuelles est à adapter. La partie fonctionnelle ne nécessite pas de retouches.
- Le langage C# (avec le framework .NET) est populaire et de nombreuses librairies et API sont disponibles pour convenir au besoin de l'outil.
- Enfin cette technologie est bien connue et maitrisée de la maitrise d'œuvre (Maxime SERRA) ce qui permettra de garantir la robustesse et la pérennité souhaitée par la maitrise d'ouvrage (Benoît FEILDEL).

V Définitions

A Fichier GPX

Ce qui sera appelé un fichier *GPX* dans le reste de ce document est un fichier portant l'extension *.gpx* et dont le contenu, écrit en XML, est normé selon le schema GPX 1.1 [4].

Ce type de fichier est le fichier qui est directement récupéré depuis l'appareil GPS qui enregistre les déplacements des personnes. Un tel fichier contient une structure XML ayant la hiérarchie suivante :

```
<gpx>
  <trk>
    <name></name>
    <desc></desc>
    <trkseg>
      <trkpt lat="47.392831" lon="0.389487">
        <ele>117.619995</ele>
        <time>2013-02-09T28:19:46Z</time>
        <speed>1.68</speed>
      </trkpt>
      ...
      <trkpt ...>
      </trkpt>
    </trkseg>
  </trk>
  ...
  <trk>
  ...
  </trk>
</gpx>
```

exemple_gpx.xml

Un grand nombre de balises et d'attributs existent dans la définition du schéma XML d'un fichier GPX. Néanmoins, seul ceux qui nous sont pertinents sont mentionnés ci-dessus.

4. GPX 1.1 Schema : *http ://www.topografix.com/gpx/1/1/gpx.xsd.*

B Trace GPS

On qualifiera de *trace GPS* (ou *trace*) une suite de coordonnées, latitudes et longitudes, sur une période données accompagnée éventuellement d'autres informations complémentaires telles que la vitesse instantanée, la date, etc.

Par analogie avec la section précédente (i.e. Fichier GPX), une trace GPS représente tous les nœuds <**trkpt**> présents dans un nœud <**trk**>, c'est à dire une suite de points qui définissent l'ensemble des déplacements réalisés par une personne.

VI Spécifications technico-fonctionnelles

Les spécifications technico-fonctionnelles qui vont être présentées dans la suite de ce document reprennent les fonctionnalités souhaitées qui ont été exprimées dans le cahier des charges mais seront découpées en sous-fonctionnalités et expliquées plus en détail.

A Import/Export (T001)

La première des fonctionnalités technique est celle de pouvoir ouvrir un fichier de données GPX et de pouvoir exporter des données après traitement.

L'application devrait pouvoir être capable d'importer les types de fichiers de données suivants :

— .GPX
— .KML *(optionnel)*

Cette liste est non-exhaustive et pourra éventuellement être complété dans l'avenir.

L'export devra théoriquement pouvoir également se faire selon les même formats d'import supportés.

B Enregistrement du travail (T002)

L'enregistrement du travail en cours consiste en la sauvegarde instantanée d'un état du travail à l'instant t. Des traitements initiés par l'utilisateur peuvent, pour une raison ou une autre, devoir être interrompus.

Ainsi, il est nécessaire de prévoir une fonctionnalité de sauvegarde et de restauration de l'état du travail en cours.

A cette fin diverses questions se posent telles que : Quelles données seront sauvegardées ? Sous quel format (fichier xml ? base de données ?) ? Cette fonctionnalité sera gérée en deux temps.

Tout d'abord, l'enregistrement du travail résidera dans la sauvegarde d'un nouveau fichier GPX incluant toutes les transformations et traitements ef-

fectués par l'utilisateur. Lors de la réouverture du programme, il pourra être proposé à l'utilisateur de reprendre ce dernier fichier.

Dans un second temps, il pourra être envisagé, si le temps le permet, d'implémenter un historique détaillé des diverses opérations effectuées afin de proposer un mécanisme de retour arrière permettant d'annuler un ou plusieurs derniers traitements. Ceci afin de fournir une meilleure expérience utilisateur.

C Traitement des traces GPS (F001)

Il s'agit dans un premier temps de pouvoir effectuer un post-traitement sur les traces GPS recueillis afin d'en éliminer les points incohérents dit "fantômes" pour avoir une traces GPS la plus propre et cohérente possible.

Dans un second temps, il faut également pouvoir détecter les points qui correspondent à un arrêt des déplacements.

Lissage (F001-01)

Le lissage d'une trace GPS va consister en l'application, sur cette dernière, d'un ou plusieurs algorithmes dans le but de réaliser plusieurs tâches :

F001-011 Tout d'abord effacer les points dit "fantômes" qui présentent une incohérence vis à vis du parcours générale. Dans ce but, diverses méthodes peuvent être envisagées.

La première consiste à parcourir la suite de coordonnées et à en éliminer toutes les positions résultant d'une accélération "improbable" par rapport à un seuil pouvant être fixé par l'utilisateur (accélération ou vitesse limite). Cette approche basique et peu couteuse (environ $O(n)$ pour n coordonnées) permet de faire une première passe qui élague quelques points réellement hors-normes.

Comme seconde passe, on peut envisager une méthode basée sur un principe de prédiction. Pour un point M donné, compte tenu de sa vitesse et du fait que l'intervalle d'enregistrement des coordonnées est compris globalement entre 1 et 3 secondes, on peut déterminer un cercle de centre M et de rayon r (distance probable parcourue en 3

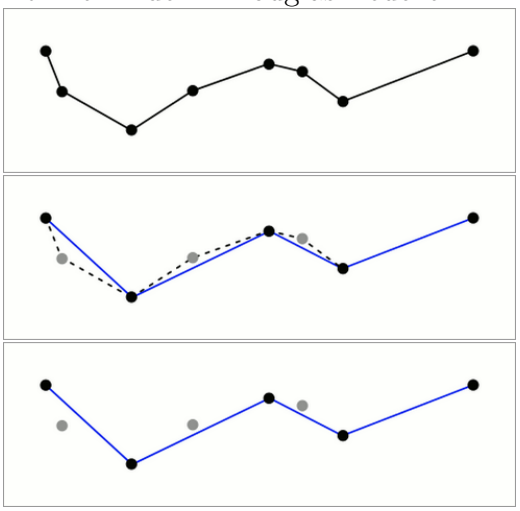
secondes en fonction de la vitesse) dans lequel devrait théoriquement se trouver le prochain point.

F001-012 Dans un second temps, il sera question de réduire le nombre de coordonnées d'une trace afin d'améliorer le temps de traitement d'affichage tout en conservant la qualité de la trace en minimisant la perte d'information.

Ceci pourra être réalisé grâce à l'algorithme de Douglas-Peucker [5].

Cet algorithme permet de simplifier le tracé de plusieurs lignes (i.e. une polyligne) par la suppression de points basée sur un critère de distance. Le fonctionnement de cet algorithme sera détaillé plus tard.

Ci-contre, un schéma expliquant la simplification d'une polyligne selon l'algorithme de Douglas-Peucker.



Amélioration (F001-02) ***(optionnel)***

L'amélioration de la trace GPS consistera à rectifier/redresser si nécessaire les coordonnées afin que le tracé corresponde à des chemins urbains ou ruraux existants.

Une solution possible à la réalisation de cette fonctionnalité est le couplage avec une base de données SIG afin d'y confronter les coordonnées d'une trace.

D Analyse des traces GPS (F002)

Il s'agit de pouvoir analyser les traces GPS dans le but d'en ressortir un certain nombre d'informations telles que la vitesse de déplacement, le type

5. Douglas-Peucker Algorithm : *http ://utpjournals.metapress.com/content/fm576770u75u7727/*

de moyen de transport ou encore les émissions de CO_2.

De ces informations peuvent également en ressortir des statistiques journalières ou hebdomadaires.

Informations primaires (F002-01)

Il s'agit d'extraire des informations dites "primaires" dans le sens où ces dernières ne présentent pas de difficulté d'extraction particulière. On pourra trouver dans cette catégorie les informations suivantes :

F002-011 Calcul de la vitesse moyenne sur le parcours
F002-012 Calcul de la distance totale parcourue
F002-013 Calcul de la durée totale du trajet
F002-014 Nombre de jours enregistrés
F002-015 Jour ou date ou heure de début et de fin selon la visualisation choisie (journalière, hebdomadaire, etc.)
F002-016 Nombre de déplacements

Informations secondaires (F002-02)

Les informations secondaires concernent les informations dont l'extraction ou bien le calcul représente des traitements plus important. On y trouve :

F002-021 Détection des points d'arrêt et de la durée de ces derniers
F002-022 ***(optionnel)*** Calcul des émissions de différents GES (Gaz à effet de serre) tel que le CO_2 par exemple en fonction des informations primaires
F002-023 ***(optionnel)*** Détection temporelle des intersections des traces de diverses personnes pour en déduire des trajets communs

Informations ternaires (F002-03)

Ces informations nécessites plus que des calculs ou de simples algorithmes afin de les extraire. Des algorithmes complexes ainsi qu'une possible base d'apprentissage et une base de connaissances pourraient être nécessaires pour inférer. Les informations concernées sont :

F002-031 Inférer sur la nature du ou des moyens de transport utilisés au sein d'un même parcours
F002-032 Inférer sur la nature du déplacement (loisir, travail...etc)

E Visualisation des traces GPS (F003)

F003-01 Lecture temporisée d'une trace

Indépendamment de tous fonds cartographiques ou indicateur visuel, il devra être possible de permettre l'affichage des traces de manière temporisée. C'est à dire une lecture progressive point par point dont la vitesse d'affichage pourra être paramétrable par l'utilisateur.

F003-02 Visualisation à différentes échelles temporelles

Il s'agit dans un premier temps de pouvoir avoir différents types de visualisation présentant les données GPS sur diverses échelles. Ces dernières sont les suivantes :

F003-011 Visualisation globale de la trace
F003-012 Visualisation journalière
F003-012 Visualisation hebdomadaire

Ces visualisations peuvent être affichées avec ou sans fond cartographique.

F003-03 Visualisation sur fond cartographique

Le but de la fonctionnalité ***F003-002*** est de pouvoir visualiser les traces (avant ou après traitement) sur une carte géographique type SIG.

Dans ce but, l'utilisation de la librairie ***GMap.NET*** semble prometteuse du fait de sont intégration simple et du support de multiples services type SIG tels que OpenStreetMap, BingMap, GoogleMap, et bien d'autres. De même, des vues satellitaires sont également disponibles.

Il pourra notamment y avoir la possibilité pour l'utilisateur de choisir quel système sera utilisé pour l'affichage de la carte géographique.

F003-04 Représentation des points d'arrêts

Les points d'arrêt précédemment détectés grâce à la fonctionnalité ***F002-022*** doivent pouvoir être représentés par des cercles dont la surface serait proportionnel à la durée de l'arrêt.

Pour se faire, les différentes techniques de détection et méthodes de représentation seront d'avantage expliciter dans le **Rapport**.

F003-05 Indicateurs visuels

Enfin, il faudra pouvoir avoir des indicateurs visuels sur cette visualisation des traces dans le but de représenter les informations extraites par la fonctionnalité F002. Par des couleurs par exemple.

Pour une trace données, différentes portions du tracé pourront avoir des couleurs différentes afin de visualiser le type de transport ou encore la vitesse de déplacement par exemple.

La possibilité sera fournie à l'utilisateur d'afficher ou masquer tout ou partie de ces indicateurs.

F003-05 Visualisation textuelles d'une trace ou "Agenda"

Le but de cette représentation est de faire apparaître une trace GPX sous la forme d'un tableau structuré chronologiquement et détaillant les déplacements d'un individu.

Typiquement, on pourra retrouver dans ce tableau les informations suivantes :

- La date et l'heure du déplacement
- Le point de départ et d'arrivée (incluant les coordonnées et pourquoi pas une adresse)
- La distance parcourue
- La vitesse moyenne sur le déplacement
- Le moyen de transport
- L'objet du déplacement (travail, étude, courses, etc.)

La liste de ces informations est non-exhaustive et pourra se voir complétée au fur et à mesure de l'avancement du projet.

VII Maquette de l'application

Voici la maquette de l'application dans sa version de départ. Cette dernière a été élaborée sur la base de plusieurs entretiens avec Benoît FEILDEL afin de fournir un squelette viable de la structure de l'interface sur laquelle démarrer.

Cette maquette va bien entendu évoluer et plusieurs allers-retours seront nécessaire pour la faire converger vers une bonne adéquation entre les fonctionnalités et l'ergonomie souhaitées. La maquette a ici uniquement pour vocation de présenter l'architecture générale souhaitée.

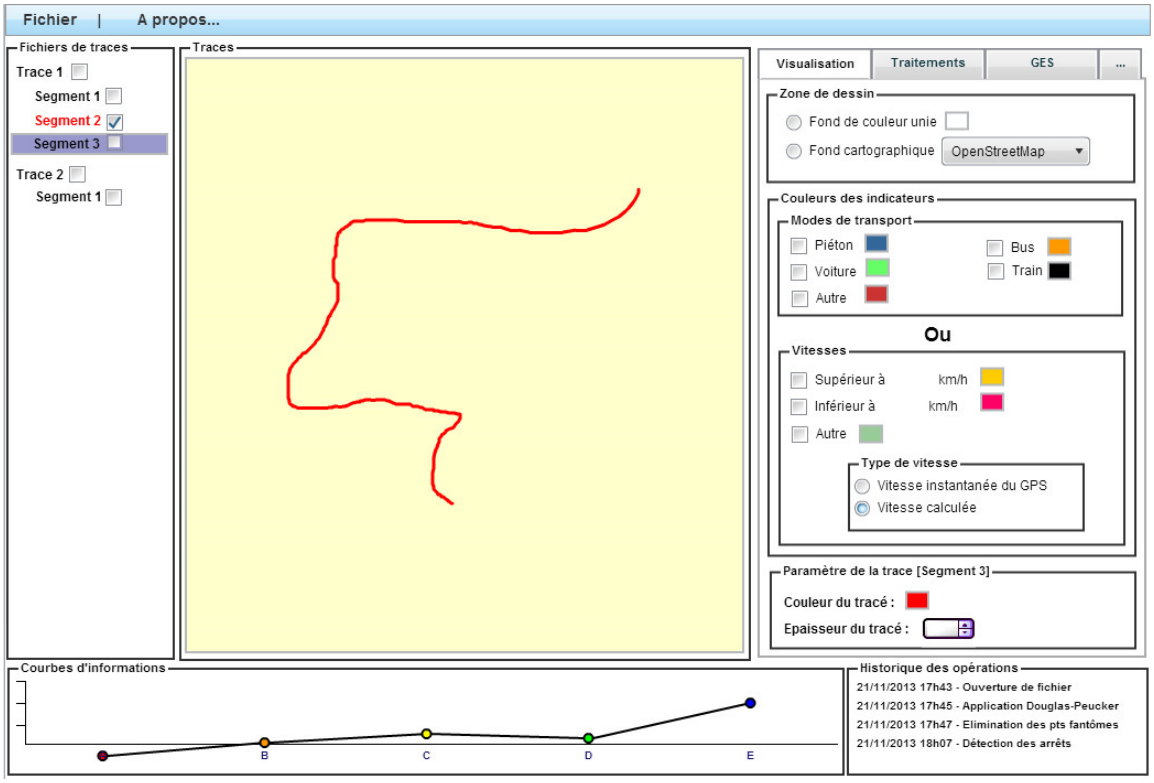


FIGURE 2 – Version béta de la maquette de l'application

MobiRev :

La Mobilité Révélée par GPS

Département Informatique
5e année
2013 - 2014

Rapport de PFE

Résumé : Le projet de fin d'étude intitulé "MobiRev : La Mobilité Révélée par GPS" consiste en l'analyse, la conception et l'élaboration d'un logiciel de traitement des traces GPS. Il doit permettre de répondre au besoin des chercheurs du département aménagement de Polytech Tours qui, dans le cadre d'une étude sur la mobilité urbaine et péri-urbaine, doivent analyser manuellement de gros volumes de données GPS. Ce présent rapport s'attache à décrire la réalisation du projet MobiRev, de la conception à la réalisation.

Mots clefs : polytech tours citeres perivia mobirev mobilité gps gpx C#

Abstract : The end of course project named "MobiRev : Mobility Revealed with GPS" consists of the analysis, the design and the development of a GPS tracks treatment software. It must help the urban design scientists of Polytech Tours to analyse large amount of GPS data in their study of urban mobility. This report aims to describe how the MobiRev project was handled, from design to development.

Keywords : polytech tours citeres perivia mobirev mobility gps gpx C#

Encadrant
Carl ESSWEIN
carl.esswein@univ-tours.fr

Étudiant
Maxime SERRA
maxime.serra@etu.univ-tours.fr

Maîtrise d'ouvrage
Benoît FEILDEL
benoit.feildel@univ-tours.fr

DI5 2013 - 2014

Université François-Rabelais, Tours