# Rapport de stage Élève Ingénieur

Emmanuel Kervizic

10 septembre 2010

# Table des matières

# Introduction

Moi, le stage

# 1 Contexte

## 1.1 Sujet du stage

Représenter le trafic aérien de la FIR de Tahiti dans le logiciel Google Earth. Le logiciel GoogleEarth permet de représenter un espace tridimensionnel, de placer, à l'intérieur du logiciel, des indicateurs tels que des marqueurs de position, des lignes, des polygones... Le logiciel GoogleEarth utilise des fichiers externes de type XML pour tracer ou représenter ces graphismes. Le format de ces fichiers est ouvert et publié par Google.

Il s'agit, à partir des traces fournies par le système de contrôle de Tahiti, d'afficher le trafic aérien circulant dans la FIR à des fins d'analyse, de vérification de trajectoire, de mesure de distance...

## 1.2 Présentation de l'environnement

### 1.2.1 DSNA/DTI

La Direction des Services de la Navigation Aérienne est chargée de rendre le service de navigation aérienne pour l'État français. A ce titre, la DSNA est responsable de rendre les services de circulation aérienne, d'information aéronautique et d'alerte sur le territoire national et ceux d'outre-mer (DOM, TOM , POM). La DSNA s'appuie sur deux directions pour exécuter cette mission :
- La Direction des opérations ou DO,
- la Direction de la Technique et de l'Innovation ou DTI.

La DO est l'acteur opérationnel du contrôle aérien tandis que la DTI est chargé du volet technique. Celui-ci consiste à réaliser ou acquérir les systèmes qui participent à l'exercice du contrôle aérien. Il s'agit de systèmes informatiques permettant d'assister le contrôleur dans ses activités, de chaînes radios pour communiquer avec les aéronefs, de systèmes de traitement de l'information météorologique...

La DTI réalise également de nombreuses études pour traiter les besoins des utilisateurs et les évolutions réglementaires. La DTI réalise le déploiement et le support opérationnel des systèmes qu'elle acquiert ou réalise.

Enfin la DTI fait viser ses systèmes, procédures et formation par l'autorité de surveillance nationale (Direction de la Sécurité de l'Aviation Civile ou DSAC).

La DTI est structurée en domaines qui sont chacun en charge de plusieurs pôles de compétences :
- Recherche & développement, R et D
- Exigences opérationnelles des systèmes, EOS
- Gestion du trafic aérien, ATM
- Communication, navigation, surveillance, CNS
- Déploiement et Support Opérationnel, DSO

Chaque pôle couvre un ensemble de fonctions et d'expertises.

Pôle ATM/VIG :
- Le pôle « Vol et information générale » (VIG) est responsable de la maîtrise d'ouvrage systèmes de traitement des plans de vol et informations générales, à ce titre, le pôle assure le suivi industriel de leur réalisation ou de leur acquisition. Le pôle VIG est également chargé de leur maintien en condition opérationnelles lorsqu'ils sont déployés.
- Le pôle ATM/VIG est notamment responsable de la maîtrise d'ouvrage de systèmes déployés en outre-mer. L'aéroport de Tahiti (Polynésie française) a récemment été modernisé avec un système entièrement acquis auprès d'un industriel, couplé à un radar dans le cadre du projet TIARE, qui s'est terminé en 2009.

## 1.3 Le site de Tahiti

### 1.3.1 Objectif contrôle aérien

Le contrôle aérien est un ensemble de services (cf. 1.3.2 page suivante) rendus par les contrôleurs aériens aux aéronefs afin d'aider à l'exécution sûre, rapide et efficace des vols. Les services rendus sont au nombre de trois, appelés « services de la navigation aérienne », dans les buts de :
- prévenir les collisions entre les aéronefs et le sol ou les véhicules d'une part, et les collisions en vol entre aéronefs d'autre part (autrefois appelés « abordages »). Il consiste aussi à accélérer et ordonner la circulation aérienne,
- de fournir les avis et renseignements utiles à l'exécution sûre et efficace du vol : informations météorologiques, information sur l'état des moyens au sol de navigation, information sur le trafic (quand le service de contrôle n'est pas assuré dans cette zone),
- de fournir un service d'alerte pour prévenir les organismes appropriés lorsque les aéronefs ont besoin de l'aide des organismes de secours et de sauvetage, et de prêter à ces organismes le concours nécessaire.

### 1.3.2 Les services de la circulation aérienne

Comme nous l'avons vu plus haut, le contrôle aérien rend plusieurs services. Nous allons voir ces services plus en détail.

#### 1.3.2.1 Le service de contrôle

Le service de contrôle est assuré dans les buts suivants :
- Prévenir les collisions entre aéronefs ou entre un aéronef et un obstacle
- Accélérer et ordonner la circulation aérienne

Le plus important reste donc la sécurité des vols. Le contrôleur s'assure que rien n'arrivera à l'aéronef pendant son vol par des causes extérieures (autre avion, obstacle), et qu'il arrivera à sa destination le plus vite possible. En outre le contrôleur est responsable de la sécurité des vols sous sa juridiction.

Les moyens qu'utilise le contrôleur pour prévenir les abordages sont la séparation (anciennement l'espacement) et l'information de trafic.
- La séparation consiste à ménager entre deux aéronefs une distance minimale, garantissant la sécurité de ces deux avions.
- L'information de trafic est une information précise sur la position d'un autre aéronef pouvant se rapprocher dangereusement. Le pilote peut ne pas voir qu'un avion se rapproche, l'information de trafic l'aide à voir, afin de permettre au pilote d'éviter l'aéronef conflictuel.

#### 1.3.2.2 Le service d'information

Le service d'information de vol est assuré sur tout le territoire français. En espace aérien contrôlé, il est assuré par le service de contrôle. Dans les espaces aériens non contrôlés, il est assuré par un organisme UIV (dans les CRNA) et SIV (dans les approches) en vol, ou AFIS sur un aérodrome. Il consiste à délivrer aux aéronefs les renseignements et avis nécessaires à l'exécution sûre et efficace du vol. Ces renseignements peuvent être (liste non exhaustive) :
- Météorologiques : conditions météo sur un terrain, présences d'orages...
- Information sur le trafic (à ne pas confondre avec l'information de trafic) : information sur un trafic connu ou inconnu, en fonction des éléments disponibles, pouvant interférer avec un aéronef.
- État des aides à la navigation
- État des équipements sol d'un terrain
- Amendements de plan de vol

– Information sur la position, aide aux pilotes perdus
– Autres...

#### 1.3.2.3 Le service d'alerte

Le service d'alerte est aussi vaste que naturel. Il consiste à répondre à tous les besoins des avions qui se disent en détresse, ou dont on peut penser qu'ils sont en détresse. Ce service recouvre des domaines très variés :
– Si un avion a déposé un plan de vol, et que le contrôle à l'arrivée a reçu confirmation qu'il a bien décollé, il doit surveiller que l'avion arrive bien à destination aux alentours de l'heure prévue, et lancer des recherches si ce n'est pas le cas.
– Si un avion ne répond plus à la radio et disparaît du radar, le contrôleur doit vérifier si l'aéronef a eu un problème et s'il s'est écrasé ou posé en urgence. Il déclenche alors les secours pour rechercher l'épave et secourir les occupants.
– Si un aéronef s'écrase sur la piste ou à proximité de l'aérodrome, il déclenche immédiatement les secours et coordonne leur action jusqu'à l'arrivée des renforts.
– Si un pilote signale avoir des problèmes avec son aéronef de nature à entraver le bon déroulement du vol, le contrôleur peut lui donner une priorité absolue à l'atterrissage en écartant tous les autres aéronefs.
– Si le contrôleur sait ou soupçonne qu'un aéronef est détourné, il prévient les autorités compétentes et leur apporte tout le secours nécessaire.

D'une manière générale, ce service est une autorisation légale à porter secours par tous les moyens à un pilote en difficulté. Tout être humain le ferait, mais le service d'alerte donne au contrôleur une justification légale pour retarder ou dérouter certains aéronefs afin de porter secours à un autre.

### 1.3.3 La zone de contrôle de Tahiti : Fir

#### 1.3.3.1 Le transport aérien

L'île de Tahiti est desservie par l'Aéroport International Tahiti Faa'a, situé à 5km au Sud-Ouest de Papeete. Inauguré en 1961, et détenu à 57% par le Territoire de la Polynésie Française10, c'est le plus important aéroport de la Polynésie française, et le seul aéroport international du territoire. Il s'agit donc de l'unique point d'entrée pour l'immense majorité des visiteurs mais également pour les habitants des autres îles de la Polynésie française.

L'aéroport assure les liaisons avec une dizaine de destinations internationales : Los Angeles, Paris, Auckland, Sydney, Tokyo, Rarotonga, Santiago, l'Île de Pâques, Nou-

mea et Honolulu10. Conscient de l'importance des liaisons aériennes internationales dans le développement économique de l'île et du pays, le gouvernement a inauguré en 1998 sa propre compagnie aérienne : Air Tahiti Nui (ATN), qui dessert aujourd'hui 5 destinations à partir de Tahiti : Paris, Los Angeles, Tokyo, Auckland, Sydney.

Concernant le réseau domestique, l'aéroport dessert l'ensemble des archipels de la Polynésie. Air Tahiti est la seule compagnie à desservir régulièrement les îles polynésiennes, assurant la liaison avec une quarantaine d'îles et d'atolls. L'île de Moorea, située à 7 minutes de vol de Tahiti est desservie par Air Moorea, une filiale de la compagnie domestique d'Air Tahiti. L'aéroport de Tahiti est la plaque tournante du trafic aérien, puisque la majorité des destinations sont uniquement desservies par l'aéroport de Tahiti. La centralisation du réseau aérien accentue donc l'attraction et l'influence de Tahiti et de l'agglomération de Papeete sur le reste des îles polynésiennes.

#### 1.3.3.2 Etendue de la FIR

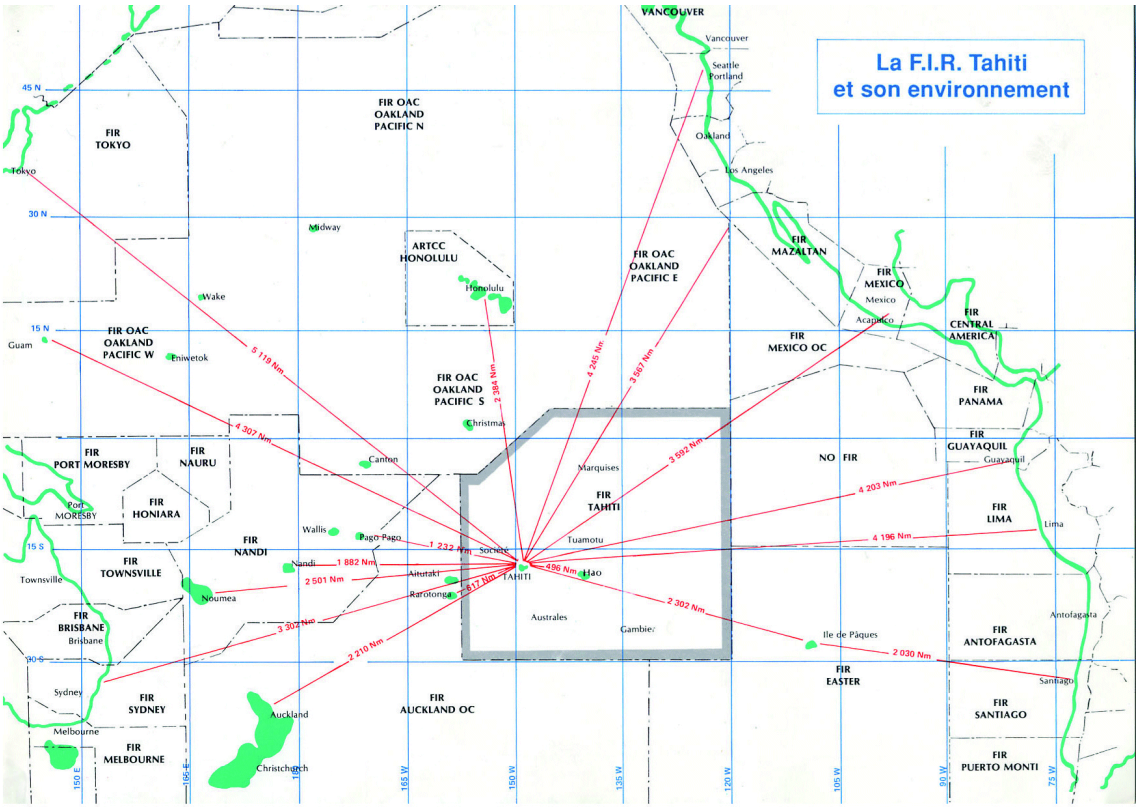


FIGURE **1.1** – FIR de Tahiti dans la région Pacifique-Sud.

La région d'information de vol de Tahiti ou « Flight Information Region » (FIR Tahiti) s'étend bien au-delà des eaux territoriales et déborde même sur l'hémisphère nord pour atteindre le parallèle 03 ° 30' Nord, soit près de 3700 km de nord au sud

et à peu près autant d'est en ouest, couvrant environ 12,5 millions de km$^2$.

Cette FIR constitue le volume au sein duquel la fourniture des services de la circulation aérienne sont assurés sous la responsabilité de l'administration française. Ces services comprennent :
- alerte et sauvetage
- information de vol
- contrôle de la circulation aérienne

La FIR Tahiti est fréquentée par différents types de trafic :
- les vols transpacifiques (entre la côte ouest des Etats-Unis et la Nouvelle-Zélande ou l'Australie)
- la desserte internationale de Tahiti (depuis et vers les Etats-Unis, la Nouvelle-Zélande, l'Australie, le Japon et le Chili)
- les vols intérieurs (desserte domestique des 47 aérodromes de Polynésie Française)

Plus de 40 contrôleurs aériens travaillent 24h/24 et 7j/7 dans la tour de contrôle de Tahiti-Faa'a.

Plus de 20 contrôleurs travaillent sur les aérodromes contrôlés des îles.

En 2006, le centre de contrôle a contrôlé 102 132 mouvements (+2,5%), dont 71477 mouvements IFR [1] et 30655 mouvements VFR [2].

#### 1.3.3.3 La zone ACI :

Une fonction de contrôle spécifique, nommée ACI [3] ou zone ACI, a été développée dans le système EUROCAT-X pour répondre à des besoins de contrôle. Il s'agit d'une zone particulière limitrophe à la FIR (cf. 1.3.3.2 page précédente) de Tahiti, dons la limite se situe à 50 miles nautiques de la FIR. La zone ACI encercle la FIR. Il est à noter que cette zone n'est pas sous la responsabilité des contrôleurs aériens français, cependant, les vols pénétrant dans cette région sont visualisés par le système Eurocat-X

Ainsi en visualisant le trafic aérien dans la zone ACI, les contrôleurs peuvent maintenir les séparations entre les aéronefs. C'est-à-dire vérifier que les vols qui sont à l'extérieur et longent la FIR de Tahiti sont séparés des vols évoluant dans cette FIR.

1. IFR : (soit, en anglais, Instrument flight rules) règles de vols aux instruments
2. VFR : Visual flight rules, nom anglais de « Vol à vue »
3. ACI : Area Common Interest, soit une zone d'intérêts commun

### 1.3.4 Le système de contrôle : TIARE

TIARE est le nom donné au projet qui a débuté en 2007 pour s'achever fin 2010. Ce projet visait à moderniser les moyens informatiques de contrôle du centre de Tahiti, de remplacer les systèmes vieillissants de visualisation du trafic (VIVO) et de gestion de plans de vol et d'informations générales (SIGMA). La DTI a fait l'acquisition de deux systèmes différents pour couvrir l'ensemble des missions dévolue aux personnels du bureau de piste et du contrôle aérien.

Les situations de contrôle auxquelles doivent face les contrôleurs sont multiples, il y en a en effet à traiter les spécificités du contrôle océanique, du contrôle d'approche et inter-îles. Le système TIARE est construit à partir de plusieurs « produits sur étagère » :

- EUROCAT-X, système en charge du traitement radar et de la gestion plans de vols.
- ATALIS, système en charge de la préparation des vols, de la gestion des NOTAM, et de la présentation d'informations générales au contrôleur tour et approche.

Les systèmes EUROCAT-X et ATALIS sont connectés au commutateur CAGOU, raccordé aux liaisons externes (RSFTA). ATALIS reçoit également des informations météorologiques en provenance du système local d'acquisition de ces données appelé CAOBS. EUROCAT-X est raccordé au radar secondaire du mont Marau et au réseau ACARS.

PREVOIR SHEMA

La zone Aci (cf. 1.3.3.3 page précédente), a été développée spécifiquement dans le système EUROCAT-X pour répondre à des besoins de contrôle.

# 2 Expression du besoin et gestion de projet

## 2.1 Expression du besoin

### 2.1.1 Les besoins initiaux

L'objectif initial était de pouvoir réaliser un logiciel banalisé et ergonomique permettant de représenter l'ensemble des données de contrôle (repères, balises, secteurs...) afin de pouvoir visualiser le trafic aérien circulant dans la FIR et la zone ACI. Les bénéfices attendus de cet outil sont :

- l'amélioration de l'analyse et de la compréhension visuelle du trafic aérien de Tahiti,
- la possibilité d'élaborer de statistiques à partir des fonctions de calcule du logiciel,
- une aide dans le travail de définition des points d'entrée dans la zone ACI que réalise le service de contrôle de Tahiti.

### 2.1.2 L'évolution des besoins

Au début de projet les besoins initiaux ont été définis. Nous verrons par la suite comment ceux-ci ont pu évoluer. Il faut noter que le client est assez dirigiste, il a déjà vu ce produit pour d'autres applications et a donc une vue global de ce qu'il souhaite en sortie. A savoir :

- Une application étant basée sur le logiciel GOOGLE EARTH.
- Python comme langage de programmation

L'objectif du choix de ces outils était aussi pour le client l'assurance de proposer à l'issue du stage une maquette complètement fonctionnelle. C'est-à-dire qu'il fallait déjouer la difficulté technique, comme représenter du trafic sur une sphère, pour se concentrer sur les besoins suscités par cet outil. En outre la durée du stage et la part consacrée à la rédaction du rapport de stage ne permettaient pas d'innover en créant un logiciel de toute pièce.

C'est pourquoi nous avons orienté notre gestion de projet vers une méthode dite agile (cf. 2.2.2 page 13). Cette méthode nous permettra de redéfinir les besoins tout au long du projet en fonction de ce qui a déjà été réalisé. Et ainsi obtenir un produit correspondant au mieux a ce que le client aurait pu espérer.

Lors du lancement du projet les besoins étaient :

- Représenter le trafic aérien déposé par les plans de vol dans la zone de contrôle de TAHITI dans GOOGLE EARTH.
- Visualiser la configuration de la plate-forme TIARE (zone de contrôle, point caractéristique ...)

Au fur et à mesure de la progression et des possibilités du logiciel, le client a affiné ses besoins et a rajouté les éléments suivants :

- Représenter le trafic aérien en fonction du temps
- Définir approximativement l'heure d'entrée de et sortie des avion dans la FIR (cf. 1.3.3.2 page 8) en fonction de leur plan de vol déposé.
- Visualiser le vol des avions en temps réel grâce aux données ADS (cf. ?? page ??).
- Visualiser le positionnement des avions estimé par le système TIARE entre deux reports ADS afin de visualiser l'interprétation des données reçue par le système.
- Différentier les type de vol en quatre catégories : Entrant, Sortant, Transit, Interne.

### 2.1.3 Les risques

Lorsque l'on a comme projet de réaliser une application qui a déjà été réalisé par le passé nous avons une base sur la-quel se référencer (en terme de méthode, de temps, de coûts). Hors sur un projet tel que le nôtre ou même aucun prototype n'a encore été réalisé le risque que cela ne fonctionne pas est très élevé.

C'est pour cela qu'une méthode de gestion de projet dite agile décrite ci-dessous (cf. 2.2.2 page suivante) à été utilisé. Cette méthode nous a permis d'avancer petit a petit afin de susciter des besoins "réalisable".

Défaut de la méthode agile : besoins sans fin, nécessité d'avoir un groupe restreint de personne, nécessité d'avoir un expert pour guider.

## 2.2 Gestion de projet

### 2.2.1 Choix de la méthode de gestion de projet

Comme nous l'avons vu précédemment, les besoins ne sont pas clairement définis dès le début. Il était donc difficile de pouvoir établir des spécifications claire affin de pouvoir réaliser un cycle en V (cf. 2.2 page 14). Nous avons donc choisis une méthodologie de gestion de projet différente de celle appliquée en temps normal à la DTI.

Cette méthodologie devait nous permettre de débuter un projet sans en connaitre

réellement l'aboutissement final tout en gardant de la rigueur et de l'organisation. Nous avons donc décidé d'utiliser une méthode dites agile [1]. Après quelque recherche et comparaison notre choix c'est orienté sur l'extreme programming (cf. 2.2.2) nous allons donc vous décrire cette méthodologie et la comparé avec le système utilisé habituellement.

### 2.2.2 L' Extreme Programming

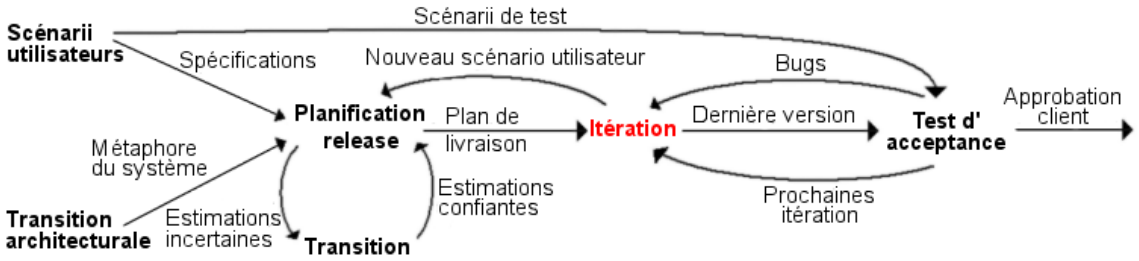


FIGURE **2.1** – Cycle de l'Exreme Programing.

L'Extreme Programming a été inventée par Kent Beck, Ward Cunningham et Ron Jeffries pendant leur travail sur un projet « C3 » de calcul des rémunérations chez Chrysler. Kent Beck, chef de projet en mars 1996 commença à affiner la méthodologie de développement utilisée sur le projet. La méthode est née officiellement en octobre 1999 avec le livre Extreme Programming Explained de Kent Beck. "Wikipedia".

Dans les méthodes traditionnelles, les besoins sont définis et souvent fixés au départ du projet, ce qui accroît les coûts ultérieurs de modifications. Extreme programming s'attache à rendre le projet plus flexible et ouvert au changement en introduisant des valeurs de base, des principes et des pratiques.

L'Extreme Programming repose sur des cycles rapides de développement (des itérations de quelques semaines voir dans notre cas quelques jours seulement) dont les étapes sont les suivantes :

- une phase d'exploration détermine les scénarios clients qui seront fournis pendant cette itération,
- la transformation des scénarios en tâches à réaliser et en tests fonctionnels,
- lorsque tous les tests fonctionnels passent, le produit est livré.

1. Les méthodes Agiles sont des groupes de pratiques pouvant s'appliquer à divers types de projets, mais se limitant plutôt actuellement aux projets de développement en informatique (conception de logiciel). Les méthodes Agiles se veulent plus pragmatiques que les méthodes traditionnelles. Elles impliquent au maximum le demandeur (client) et permettent une grande réactivité à ses demandes. Elles visent la satisfaction réelle du besoin du client et non les termes d'un contrat de développement.

Lorsqu'une tâche est terminée, les modifications sont immédiatement intégrées dans le produit complet. On évite ainsi la surcharge de travail liée à l'intégration de tous les éléments avant la livraison. Les tests facilitent grandement cette intégration : quand tous les tests passent, l'intégration est terminée.

Le cycle se répète tant que le client peut fournir des scénarios à livrer (cf. Fig. 2.1 page précédente). Généralement le cycle de la première livraison se caractérise par sa durée et le volume important de fonctionnalités embarquées. Après la première mise en production, les itérations peuvent devenir plus courtes (par exemple la séparation des plans de vol en catégories tel que : transit, interne ...)

Pour résumé, cette méthode nous amène à réaliser la boucle suivante :
- Analyse du besoin.
- Expression des spécifications
- Réalisation technique
- Test de la réalisation
- Revue logicielle (validations qui permettront de faire évoluer le produit)

### 2.2.3 Le cycle en V

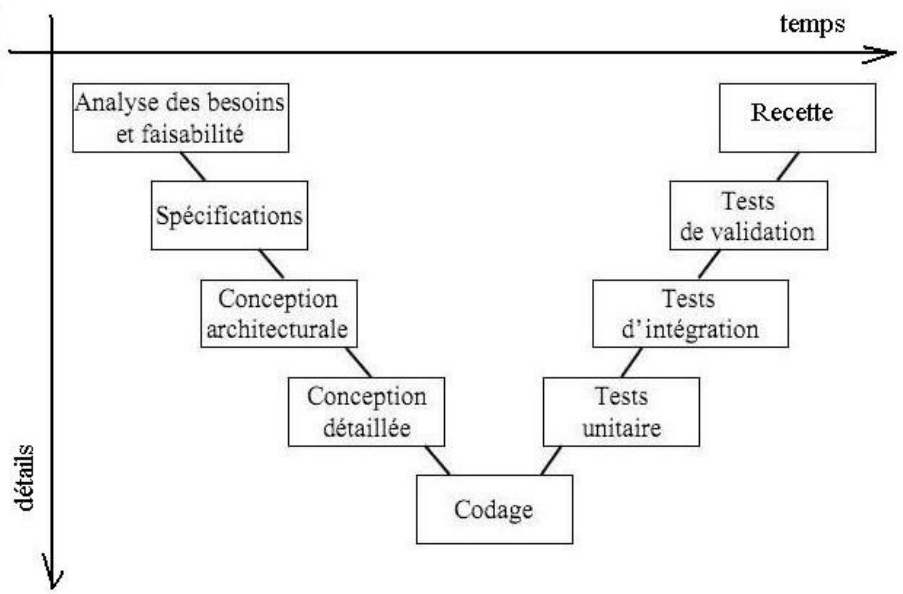


FIGURE **2.2** – Les phases à travers le temps et le niveau de détails.

Le modèle du cycle en V est un modèle conceptuel de gestion de projet imaginé suite au problème de réactivité du modèle en cascade. Il permet, en cas d'anomalie, de limiter un retour aux étapes précédentes. Les phases (cf. Fig. 2.2) de la partie montante doivent renvoyer de l'information sur les phases en vis-à-vis lorsque des défauts sont détectés, afin d'améliorer le logiciel.

Le cycle en V est devenu un standard de l'Industrie logicielle depuis les années 1980 et depuis l'apparition de l'Ingénierie des Systèmes est devenu un standard conceptuel dans tous les domaines de l'Industrie. Le monde du logiciel ayant de fait pris un peu d'avance en termes de maturité, on trouvera dans la bibliographie courante souvent des références au monde du logiciel qui pourront s'appliquer au système.

Les étapes qui constituent cette méthode sont les suivantes :
- Analyse des besoins et faisabilité
- Spécification logicielle
- Conception architecturale
- Conception détaillée
- Codage
- Test unitaire
- Test d'intégration
- Test de validation (Recette Usine, Validation Usine - VAU)
- Recette (Vérification d'Aptitude au Bon Fonctionnement - VABF)

On voit bien que dans notre cas, avec des besoins indefini, il est impensable d'appliquée une telle méthodologie sans engendrer le rique que le logiciel final ne marche pas ou ne reponde pas aux attentes du client.

### 2.2.4 Les avantages et inconvénients

Les avantage de cette méthode, dans cette situation, serrons :
- Enrichir le produit à chaque itération du cycle. Ci le logiciel marche le client peut visualiser immédiatement les besoins qui étaient superficiels, dont il n'avait pas réellement besoins, et au contraire découvrir de nouveaux besoins réellement utiles.
- Rediriger rapidement la conduite du projet. Si le client souhaite rediriger son projet, ceci peut être fait dans les meilleurs délais (changement d'objectifs ou de priorités)

Cette méthode implique tout de même un certain nombre d'inconvénient tel que :
- Le client doit être disponible afin de faire avancer le projet. Chaque validation est vue avec le client et c'est celui-ci qui donne les nouveaux besoins. Ce qui implique que ci celui-ci n'est pas disponible, le projet peut vite être bloqué.
- Le projet peut vite dériver. Ce type de méthode requiert des personnes compétentes, aussi bien au niveau Maitre d'ouvrage, qu'au niveau maitre d'œuvre. Il est facile de s'égarer c'est pourquoi une organisation et une rigueur doivent être entretenues tout au long du projet.

– Lors du début du projet, on manque cruellement de spécifications. On se lance alors dans le développement sans analyse complète.

# 3 Réalisation technique

## 3.1 Architecture du logiciel

### 3.1.1 L'interface Google Erath

Google Earth dispose d'une interface graphique qui sera mise à profit pour :
- représenter les points remarquables (points nommés, points de coordination, etc.),
- représenter les espaces de contrôle,
- représenter le trafic aérien.

Ces données sont, soit statiques, soit dynamiques.

**Statiques :** affichage de points fixes et affichage des espaces ou trajectoire plan de vol.

**Dynamique :** représentation du trafic aérien en fonction de coordonnées mises à jour et en fonction du temps.

### 3.1.2 Gestion de l'affichage

Google Earth peut être enrichi de données externes via un fichier descriptif de données (Kmz).

Ce fichier Kmz n'est autre qu'un fichier Zip compriment un fichier "doc.kml" ainsi que les fichiers vers les quel il pointe. Ce fichier "doc.kml" a pour but de regrouper les fichier à ouvrir (comme le ferait une liste de lecture pour les fichiers Mp3). Il indique donc à Google Earth de charger les fichiers suivant :

**CharacteristicsPoints.kml :** Le fichier contenant les points remarquables

**Fir.kml :** pour les zones de contrôle et la zone Aci

**Routes.kml :** regroupe toutes les routes définies

**Fpl.kml :** affiche les plans de vol déposés

**Ads.kml :** affiche le trafic aérien réel recu par l'Adsc

Les fichier KML sont fabriqués à partir des fichiers de configuration et de traces définis dans le système EurocatX.

Le système EurocatX est configuré au moyen de fichiers de configuration statique. Ces fichiers seront «parsés» pour fabriquer les fichiers : "CharacteristicsPoints.kml", "Fir.kml" et "Ads.kml".

Ce système est également constitué de fichiers de traces qui log les informations de vol. Ces fichiers seront «parsés» pour fabriquer les fichiers : "Ads.kml" et "Fpl.kml".

### 3.1.3 Les modules

Au final le logiciel se compose des modules suivants :

**Manu :** Fichier principal, peut être considéré comme l'exécuteur. (cf. ?? page ??)

**modules/Ads :** Met en mémoire les information sur les report ADS. (cf. ?? page ??)

**modules/Aoi :** Permet de définir tout les volume utilisé pour concevoir les zone de contrôles. (cf. ?? page ??)

**modules/CharacteristicPoints :** Met en mémoire tous les points remarquables disponible sur le système. (cf. ?? page ??)

**modules/Convertion :** Regroupe plusieurs fonctions utiliser pour convertir des donnée (ex : utilisé pour convertir les coordonnées). (cf. ?? page ??)

**modules/Fdp :** définit et met en mémoire toutes les zone de contrôles. (cf. ?? page ??)

**modules/Fpl :** définit et mets en mémoire les plans de vol. (cf. ?? page ??)

**modues/GetOfFiles :** Coordonne la récupération des donnée, c'est lui qui va chercher la configuration et lance les modules tel que Aoi, Fdp ou encore Fpl. (cf. ?? page ??)

**modules/KML :** Ce module est utilisé pour mettre en forme les fichier KML à l'aide des données reçues en entrée (par exemple pour un point il reçois ça description, ces coordonnées, son nom ...). (cf. ?? page ??)

**modules/MakeKML :** C'est le module qui exploite toutes les données en mémoire et crée les fichiers KML. (cf. ?? page ??)

**modules/MakeKMZ :** Récupère les fichiers KML pour les regrouper en un fichier KMZ plus maniable. (cf. ?? page ??)

**modules/Routes :** définit et mets en mémoire les routes definies. (cf. ?? page ??)

**modules/usualFonction :** Regroupe plusieurs fonction régulièrement utilisées. (cf. ?? page ??)

Chaque modules est décrit avec plus de précisions en annexe.

Le fichier principal (Manu.py) se lance à partir de la ligne de commande : "Python Manu.py" dans un système ou Python est installé et correctement configuré.

Les fichiers ".asf" contenant la configuration de l'eurocatx sont placés dans le répertoire "SurcesAsf" alors que les fichiers contenant les traces sont dans le répertoire "Sources".

Tout les modules sont appelé automatiquement, se qui signifie qu'une foi le fichier de configuration renseigner et le fichier principal lancé, aucune action n'est nécessaire jusqu'a la réception du fichier KMZ. Il n'y aura donc plus qu'a lancer se fichier en l'ouvrant depuis GOOGLE EARTH.

## 3.2 Le contexte technique opérationnel

### 3.2.1 EUROCATX

Il faut bien comprendre comment marche le système afin de bien visualiser d'où proviennent les informations. Comme décrit grossièrement dans le schéma (cf. Fig. 3.1), EUROCATX récupère les informations sur les plans de vol par l'intermédiaire de CA-

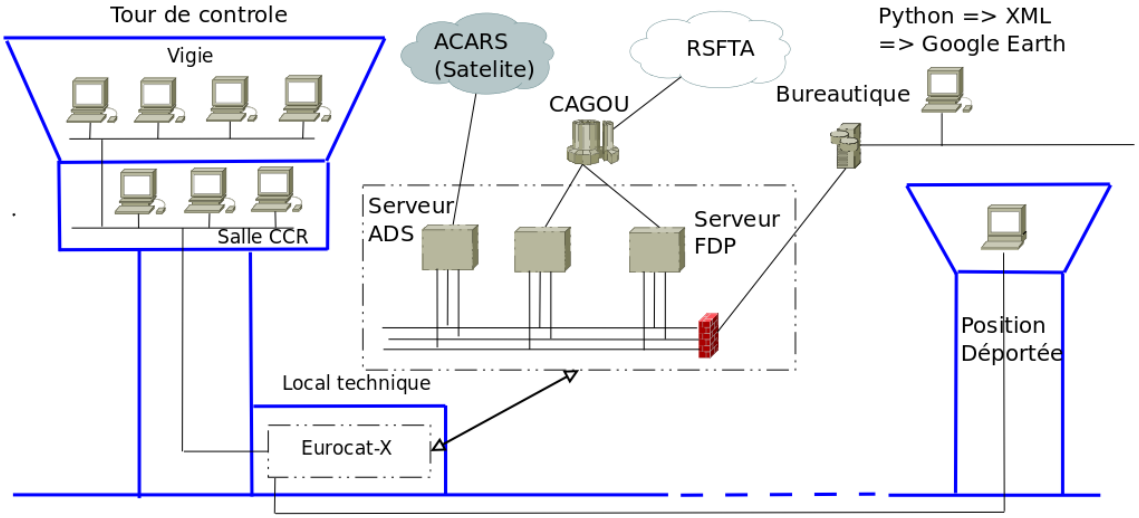


FIGURE **3.1** – Schématisation du système EUROCATX au niveau des tours de contrôle

GOU[1]. Il récupère aussi le positionnement émis par l'avion à l'aide de la transmission Satellite, VHF[2] ou des données radars lors de son approche. Le système EUROCATX donne un accès à la bureautique protégé par un par-feu (FireWall) afin de rendre disponible sur ce réseau un certain nombre d'information. Dans notre cas nous y récupérerons :

- toutes les données de configuration du système tels que les nom et coordonnées des balise référencée, la position des zone de contrôle et des zone ACI ou encore les route utilisée pour décrire les plans de vols.
- Les fichiers de log du Commutateur CAGOU afin de pouvoir exploiter les plans de vol reçus par le réseau RSFTA.

1. CAGOU : nom donné au commutateur RSFTA
2. VHF : Very High Frequency, soit une bande radio de très haute fréquence

– Tous les reports ADS reçu par satellite et traité par le système.

Le système envoyé les informations récoltées et celle calculées au visues[3] situées dans la tour de contrôle au niveau de la Vigie ou de la salle CCR ainsi que de la position déportée à MOREA.

Les données seront donc récupérée dans les fichiers ".asf" pour tous ce qui est de la configuration du système, dans les fichier du FDP pour les plans de vol et dans les fichiers du serveur ADS pour les reports ainsi que pour la position calculée des aéronefs.

### 3.2.2 Le domaine de l'aviation

Il m'a aussi été nécessaire de prendre connaissance de touts les termes, unité, convention et j'en passe utilisé dans le domaine aéronautique.

#### 3.2.2.1 Les coordonnées et unités :

Tout d'abord est vite venu le problème de conversion de coordonnées, J'ai donc du revoir les conversions de coordonnées sphériques ainsi que les conversions de distances. J'ai également du, comme expliquée ci dessous (cf. 3.5.2 page 27) me remémoré les solutions de calcule du point d'intersection de deux arc de cercle en coordonnées sphériques.

#### 3.2.2.2 Convention :

Plusieurs conventions on du être acquise comme celle utilisé par le système TIARE pour décrire les report ADS ou encre celle utilisée par les compagnies pour le dépôt de plan de vole. NE PAS OUBLIER DE FAIRE RÉF AU DOCUMENT 4444 ...

## 3.3 Base de travail

### 3.3.1 Le langage Python

#### 3.3.1.1 Bien coder :

Afin de pouvoir apprendre les bonne pratique de la programmation Python j'ai lu un livre intitulé "Programmation Python, conception et Optimisation"[5]. Celui-ci

3. Visue : Nom pour décrire les ordinateur utilisés pour visualiser les données de contrôles

m'a permit de pouvoir d'une part revoir ce qui avait été appliquer lors de mes études et d'autre part avoir une vue global sur le langage et ainsi pouvoir prendre du recule lors du codage.

Celui ci m'a par exemple appris le nouveau style de programmation qui part du principe que chaque nouvel objet définit est basé sur un Objet existant, et que par la même occasion tout en python était Objet (même une simple variable booléenne). Ou encore la manière de vérifier si un objet était faux, égale à 0 ou encore une chaîne vide simplement en demandant si il existait (ex : `"if x != 0:"` devient `"if not x:"`)

#### 3.3.1.2 Utiliser les expressions régulières :

L'apprentissage de l'utilisation des expressions régulières [4], m'a été grandement facilité garce au site : http://www.dsimb.inserm.fr/[2] et a la documentation en ligne de Python[4]. Il s'est avéré après apprentissage que ces expression régulière aurons grandement facilité la faisabilité du projet.

#### 3.3.1.3 L'optimisation :

Je pourrais cité un passage du livre[5] qui dit :

> Fourni dès le départ avec des modules de tests, Python est un langage agile. Le terme agile est originellement issu de la méthodologie de programmation agile (Beck et Al.), très proche de la programmation itérative. Cette méthodologie, qui réduit les risques liés à la conception de logiciels, introduit entre autres des principes de tests continus du code. Vincent Lozano.

En effet il m'a été rapidement nécessaire de réalisé des test, aussi bien pour vérifier que mon code était valide que pour vérifier que celui-ci s'exécutait normalement. Il c'est avéré à plusieurs reprises que certaines parties de mon code étaient très gourmandes en processus. L'apprentissage de fonctions de test de code, tel que le module hotshot décrit plus tard (cf. 3.5.3 page 29), m'a été rapidement nécessaire.

### 3.3.2 Google Earth

Google Earth est un logiciel, propriété de la société Google, permettant une visualisation de la terre en 3 dimensions avec un assemblage de photographies aé-

4. Une expression régulière est en informatique une chaîne de caractères que l'on appelle parfois un motif et qui décrit un ensemble de chaînes de caractères possibles selon une syntaxe précise.

riennes ou satellites. Ce logiciel donne la possibilité de configurer un environnement, ajouter des lignes, des points ou encore des polygone en 3D en passent par des fichier de configuration au format KML [5].

Ce format, qui repose sur le XML [6], a l'avantage d'être simple à manipuler. Ça sémantique est définie sur le de google (cf. Bibliographie [3])

## 3.4 Le programme réalisé et ses fonctions

### 3.4.1 Le fonctionnement

**La configuration :** Le programme réalisé ne possède pas encore d'interface (IHM) graphique. Il est donc nécessaire de configurer les option a l'aide d'un fichier de configuration (cf. annexe ?? page ??). Nous pourrons régler par l'intermédiaire de celui-ci :

- Les fichiers Kml à recréer ou non, se qui est utile afin de ne pas avoir à recréer des fichiers statique (tel que la position des point caractéristique ou encore des zones de contrôles) a chaque utilisation tout en laissant a l'utilisateur la possibilité de les mettre a jour simplement.
- Les différant styles et couleurs.
- L'emplacement des fichiers de configuration.
- les description et noms appliqué à chaque catégorie.

**L'exécution :** Le fichier de configuration renseigné, le programme peut être lancé. Il est possible de le lancer par l'intermédiaire d'un Shell [7], par l'intermédiaire de l'interface Python ou encore en direct si les informations pour gérer et lancer les fichiers Python ont été renseignées dans le système d'exploitation.

**Le résultat :** L'exécution du programme réalise une suite d'action :

1. Lire le fichier de configuration afin de déterminer les action à effectuer.
2. Lire les fichiers de configuration du système TIARE affin de récupérer toutes les variable nécessaire sous forme d'objet [8] (ex : points caractéristique ...)

5. KML : Keyhole Markup Language, est un format de fichier et de grammaire XML pour la modélisation et le stockage de caractéristiques géographiques comme les points, les lignes, les images, les polygones et les modèles pour l'affichage dans GOOGLE EARTH, dans GOOGLE MAPS et dans d'autres applications.

6. XML : Extensible Markup Language («langage extensible de balisage»), est un langage informatique de balisage générique.

7. Shell : Interface en lignes de commandes

8. Objet : structure de données valuées et cachées qui répond à un ensemble de messages. Cette

3. Lire les fichiers de log afin de créer des objets tel que les plans de vol ou encore les reports ADS. Ces objet sont créer non seulement a partir de ses fichiers de log mais aussi a partir des objets créer précédemment (ex : les point des plan de vol désigné par un nom sont convertis en coordonnées à l'aide des points caractéristiques).
4. Créer les fichiers KML désigné dans le fichier de configuration à l'aide des objets instancié.
5. Créer un fichier KMZ a l'aide de tout les fichiers KML afin d'avoir un fichier compact et facile a transporter.

### 3.4.2 L'exploitation dans GOOGLE EARTH

L'execution du programme retourne en résultat un fichiers KMZ. C'est ce fichier qui est utiliser pour éxploiter les données dans GOOGLE EARTH. Pour cela il suffit d'ouvrir le fichier à l'aide de ce logiciel.

Nous allons vous présenter quelques exemple d'utilisation de ce logiciel.

#### 3.4.2.1 Vue d'ensemble

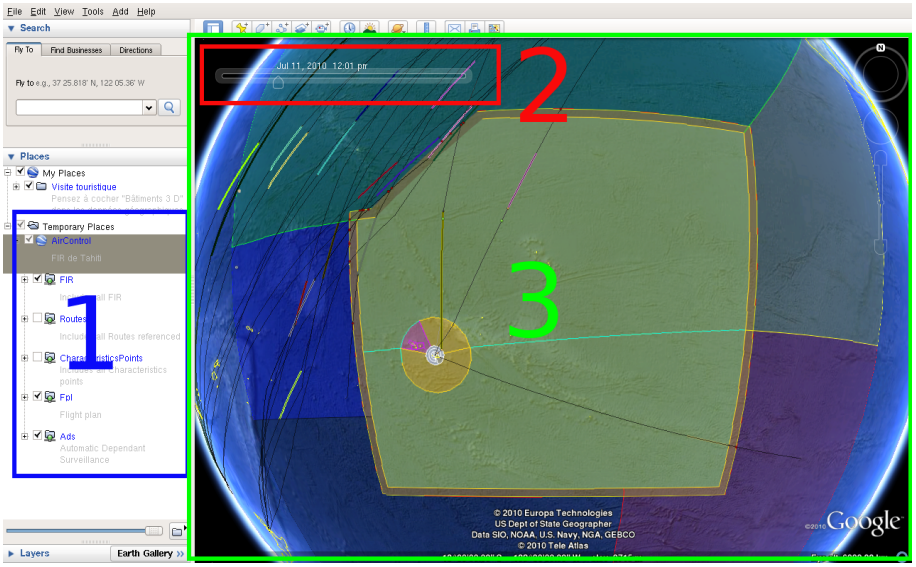


FIGURE **3.2** – Vue d'ensemble du trafic dans GOOGLE EARTH

Lors de l'ouverture du fichier la vu est centrée sur la zone de contrôle de Tahiti. Comme vous pouvez le voir (cf. Fig. 3.2) nous pouvons distinguer trois zones dans le logiciel :

structure de données définit son état tandis que l'ensemble des messages qu'il comprend décrit son comportement

**La zone de sélection :** Cette zone, numéroté 1 sur la figure, sert à sélectionner les éléments à afficher ou non.

Chaque groupe d'éléments est représenté par un dossiers. Ainsi il serra plus facile de sélectionner un groupe tel que les routes. Il sera aussi simple de dé-sélectionner un groupe (par exemple groupe Fpl qui contient tout les plans de vol) et de re-sélectionner un seul élément du groupe afin de le visualiser séparément (ex : le plan de vol d'un avion précis).

**L'animation temporelle :** Sur notre figure cet outil est représenté par le numéro 2. Nous pourrons grâce a lui visualiser l'évolution du trafic dans le temps. Les principale option utile a notre cas seront :

- La sélection d'une date et une heure précise afin de visualiser ou devrait se trouver un avion ou encore avoir une vue de tout les vole en cours à cette heure.
- Un créneau compris entre deux date et heure. Cette option nous permettra de visualiser un vol sur un partie de son parcours afin d'avoir une vue un peut plus globale. Nous l'avons utilisé lors de l'exemple suivant (cf. Fig. 3.3 page suivante) afin de mieux visualiser les points estimé par le système TIARE.

**La vue :** Cette zone, numérotée 3, nous permet visualisée notre sélection configurée à l'aide des deux zones citées précédemment.

#### 3.4.2.2 Exploitation des données

Nous avons pris en exemple un zoom sur un plan de vol qui a été dévié de son sa trajectoire initiale (son plan de vol). Nous pouvons donc apercevoir sur cette figure (cf. Fig. 3.3 page suivante) :

**Les lignes noires** Ces ligne représente les plans de vol des avions en cours a la date et l'heure sélectionnée. Ci ceux-ci ne sont pas recouvert par une ligne de couleur plus épaisse, cela signifie que l'avion n'est pas censé se trouver a cet endroit a l'instant défini.

**La ligne verte :** Cette ligne représente le plan de vol déposé. Chaque segment de cette ligne représente ou peut se situer l'avion a l'instant donné.

**La ligne blanche :** Cette ligne représente la trajectoire réellement effectuée par l'avion. Cette ligne est définie par les report ADS.

**Les points verts :** Ces points représente les report ADS reçu par le système TIARE. Lorsque l'on clic sur l'un de ces point il est possible de voir ça description qui contient le message émis par l'avion pour définir ce point.

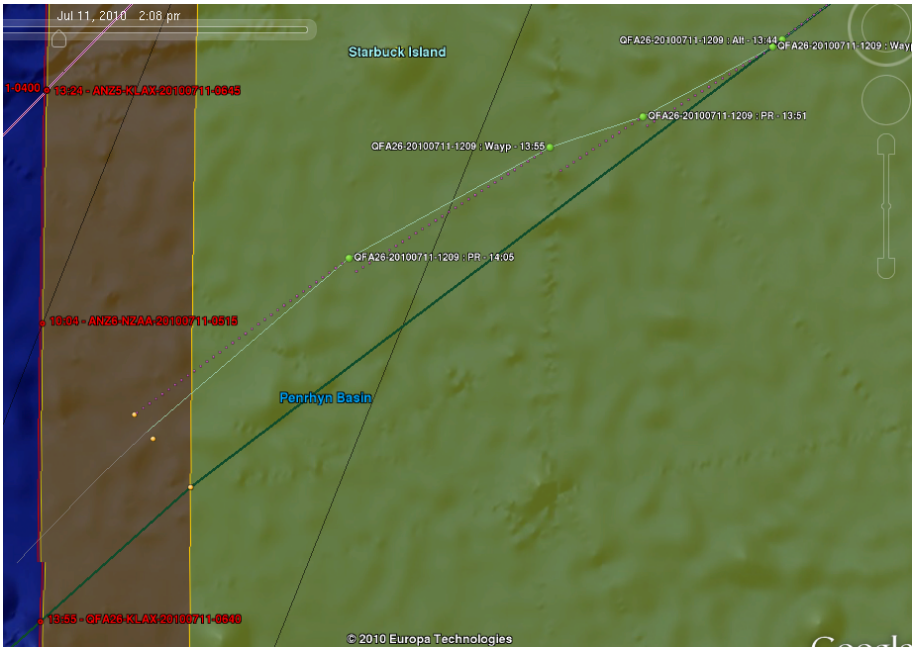


FIGURE **3.3** – Zoom su la déviation de la trajectoire d'un vol par rapport a son plan de vol déposé

**Les points roses :** Ils représentent les points estimés par le système TIARE. Dans ce cas précis nous constatons que ces points ne correspondent pas avec la trajectoire réellement réalisée. Cette différence peut s'expliquer par le fait que le système utilise les point de reports suivant estimé par l'avion pour définir la position actuel de l'avion.

**Les points oranges :** Ces point représentent les reports suivant estimés (next report) par l'avion cité précédemment.

**Les points rouges :** Ces points représente l'heure d'entrée, estimé par le programme, dans la zone ACI (cf. 1.3.3.3 page 9).

## 3.5 Problèmes techniques rencontrés et solution apportées

Comme dans tout projet il y a une multitude de problèmes à résoudre. Nous verrons dans cette partie quelques exemples de ces problèmes rencontrés ainsi que la manière dont ils ont été résolus. Cette liste reste bien entendu exhaustive au regard de tous les petits problèmes auxquels nous avons du faire face.

### 3.5.1 Gestion des erreurs

**Problématique :** Le premier problème que nous avons rencontré a été celui de la gestion des erreurs. En effet, de la première mise en route du logiciel jusqu'à la fin du stage des erreurs ont du être gérées. Deux types d'erreurs sont revenues :

- Le premier type d'erreur était par exemple une réaction in attendue du logiciel, On pourrait prendre en exemple la conversion de coordonnées reçue en Système sexagésimal[9] en coordonnées décimales utilisées dans les fichiers KML 5 page 22, qui lors des premiers tests donnait des donnée erronées.
- Le deuxième type était celui du aux erreurs contenues dans les fichiers de log utilisés pour récupérer les informations. Ces erreur faisaient effet boule de neige et venait se répercuter dans le fonctionnement du logiciel.

**Résolution :** La solution au premier problème a été de mettre en place des test a chaque fonction implémenter ou après avoir réaliser chaque objectif fixé. On appel cette méthode le test continu du code. Grâce à cela nous allons pouvoir déterminer plus rapidement lors d'une erreur futur d'où provient celle-ci. Une méthode simple de la mette en place est de définir un test a réaliser pour valider la fonction ou le code. On détermine donc quel réaction doit avoir un fonction pour un environnement donné et l'on vérifie si le résultat corresponds bien avec celui espéré. (Ex : on a la coordonnée 4530N10045E qui correspond a 45°30' Nord 100°45' Est. On envoi cette variable dans la fonction de conversion et l'on vérifie que le résultat retourné est bien en décimal : 45,5° en latitude et -100,75 en longitude). Si le résultat est correct la fonction ou le morceau de code est validé, sinon il doit être corrigé.

La solution du deuxième problème a été dans un premier temps d'afficher chaque erreur dans la console, mais cela est vite devenu trop compliqué du fait que la console ne retient par défaut qu'un nombre limité de ligne en mémoire et que les ligne trop ancienne sont simplement effacée. On a donc mis en place un système de log permettant, en plus d'avoir accès au information les plus ancienne, de pouvoir l'exploiter ares avoir fermé la console, effectuer des recherche a l'intérieur et tout avantage que peut apporter un fichier texte. Pour les dernière version de log, celles-ci sont crées avec des information relative au type d'erreur et l'emplacement de l'erreur dans le fichier source, le tout enregistrées dans un fichier comprenant la date et l'heure actuel dans le nom afin de pouvoir les différencier de chaque exécution du logiciel.

9. (Système sexagésimal : Degrés ( ° ) Minutes ( ' ) Secondes (" ))

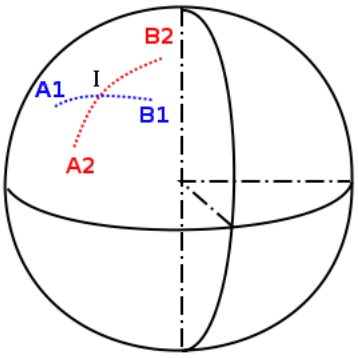


FIGURE **3.4** – Représentation grossière de l'intersection de deux arc de cercle respectivement formé par la trajectoire la plus courte entre deux points situé sur le Globe terrestre

### 3.5.2 Intersection entre plans de vol et zone ACI

**Problématique :** Afin de déterminer l'heure d'entrée approximative des avions dans la zone ACI (cf. 1.3.3.3 page 9) en fonction de leur plan de vol déposé Il est nécessaire de déterminer le point d'intersection entre leur plan de vol et la zone ACI. En théorie cela paraît simple, il suffit de prendre chaque portion du trajet du plan de vol composé de deux point et formant une droite, et de déterminer si cette droite coupe chaque droite composant la zone ACI. Dans la pratique il c'est avéré que cela était un peu plus compliqué, en effet ces droites sont en réalité des arcs de cercles qui sont composé de deux extrémités définies par des points en coordonnées sphériques (cf. schéma fig. 3.4).

**Résolution :** Étant donné que j'ai effectué un BTS avant d'intégrer l'EIGSI[10], les notion de coordonnées sphérique ne me sont que peut familière. Après avoir en vainc cherché sur internet ainsi que dans mon entourage (maître de stage, collègues de travail) je me suis replié sur un forum de mathématique sur le quel j'ai déposé un sujet explicitant le problème (adresse, cf. bibliographie [1]). Une personne nous a donnée une solution qui, après connaissance, semble tellement simple qu'on se demande pourquoi personne n'y a pensés. Cette solution consiste a déterminer les plan défini par les deux points aux extrémités de chaque arc et par le centre de la terre (ainsi nous avons forcement la courbe qu'a suivi l'avion sur ce plan). Il faut ensuite déterminer la normal a chacun des plan pour en déduire la droite d'intersection de ces plan (passant par le centre de la sphère). Une foi cette droite acquise il faut définir sont vecteur norme et le convertir en coordonnée sphérique. Ce qui nous donne un des point d'intersection de la droite avec la sphère, l'autre étant situé par définition à l'opposé.

10. EIGSI : École d'Ingénieurs en Génie des Systèmes Industriel située à La Rochelle

Une démonstration valant amplement un long discours, et a titre informatif, voici ce que cela donne en résolution mathématique. Pour cet exemple nous avons deux arcs représentent 2 trajectoires définies chacune par 2 points A et B (cf. Fig. 3.4 page précédente). Chaque pour sera défini par une latitude et une longitude.

Nous avons donc :
- $lat_A$ la latitude de A
- $long_A$ la longitude de A
- $(x_A, y_A, z_A)$ les coordonnées cartésiennes de A
- $I_1$ le point d'intersection n° 1
- $I_2$ le point d'intersection n° 2

Il faut tout d'abord convertir les coordonnées sphérique en vecteur de coordonnées cartésiennes pour $A$ et $B$ :

$$A = \begin{cases} x_A &= cos(lat) \times cos(long) \\ y_D &= cos(lat_A) \times sin(long_A) \\ z_A &= sin(lat_A) \end{cases}$$

Il faut ensuite déterminer le plan passant par $O$, $A$ et $B$ ayant alors pour équation :

$$ax + by + cz = 0$$

où

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} x_A \\ y_A \\ z_A \end{pmatrix} \wedge \begin{pmatrix} x_B \\ y_B \\ z_B \end{pmatrix}$$

c'est à dire

$$\begin{cases} a = y_A z_B - z_A y_B \\ b = z_A x_B - x_A z_B \\ c = x_A y_B - y_A x_B \end{cases}$$

L'intersection des deux plans de coordonnées $(a, b, c)$ et (a',b',c') contient le point O, mais aussi le point $P$ de coordonnées $(x_P, y_P, z_P)$ tel que :

$$\begin{pmatrix} x_P \\ y_P \\ z_P \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \wedge \begin{pmatrix} a' \\ b' \\ c' \end{pmatrix}$$

P n'étant pas forcément sur la sphère, il faut trouver un point de la droite $(OP)$ sur cette sphère. Pour cela il suffit de diviser les 3 coordonnées de P par la norme de $\overrightarrow{OP}$ :

$$I_1 = \begin{cases} x_P / \sqrt{x_P^2 + y_P^2 + z_P^2} \\ x_P / \sqrt{y_P^2 + y_P^2 + z_P^2} \\ x_P / \sqrt{z_P^2 + y_P^2 + z_P^2} \end{cases}$$

nous avons donc $I_1$ et son opposé $I_2$, il nous reste donc plus qu'a vérifier si chacun de ces points appartient à un des 2 arcs.

Vous trouverez le code Python correspondant à ces calcule dans la fonction : "verifyIntersection (line, point) :" du module "ususalFonction.py" disponible en annexe ?? page ??

### 3.5.3 Performance du logiciel

**Problématique :** Les premiers tests du logiciel ce sont déroulé sur un nombre limité de fichiers (représenté par un nombre limité d'heure de vol), ce affin de pouvoir les valider rapidement. Lors de l'apparition de fichiers plus volumineux (plus de 300Mo de donnée en entrée, environ 10% en sortie) c'est posé le problème de performance. Avant optimisation l'ordinateur moulinais des heures avant de pouvoir sortir un fichier. Il a donc fallût optimiser le code afin d'alléger le programme en ressources.

**Résolution :** En cherchant des conseils dans des forum d'informatique ainsi que dans le livre cité précédemment (cf. bibliographie [5], nous avons découvert que Python était un langage orienter par les test et qu'il disposait donc de librairies spécialement conçues pour déterminer les point bloquant d'un programme et les fonctions appelées les plus gourmandes.

La fonction retenue pour repérer ce qui est appelé en anglais les Bottleneck [11] est la fonction "hotshot" qui à pour but d'analyser un programme dans sa totalité en indiquant notamment les ressources utilisées par chaque fonction appelée. Pour visualiser ce que donne le résultat d'une analyse veuillez vous reporter a la figure 3.5 page suivante.

Les bottlenecks repérés, une réécriture des parties bloquantes à du être effectuée. Cette analyse nous a permis de réduire les ressources et donc le temps d'exécution du logiciel de plus de 80%.

11. Bottlneck : (goulot d'étranglement) point d'un système limitant les performances globales, et pouvant avoir un effet sur les temps de traitement et de réponse.

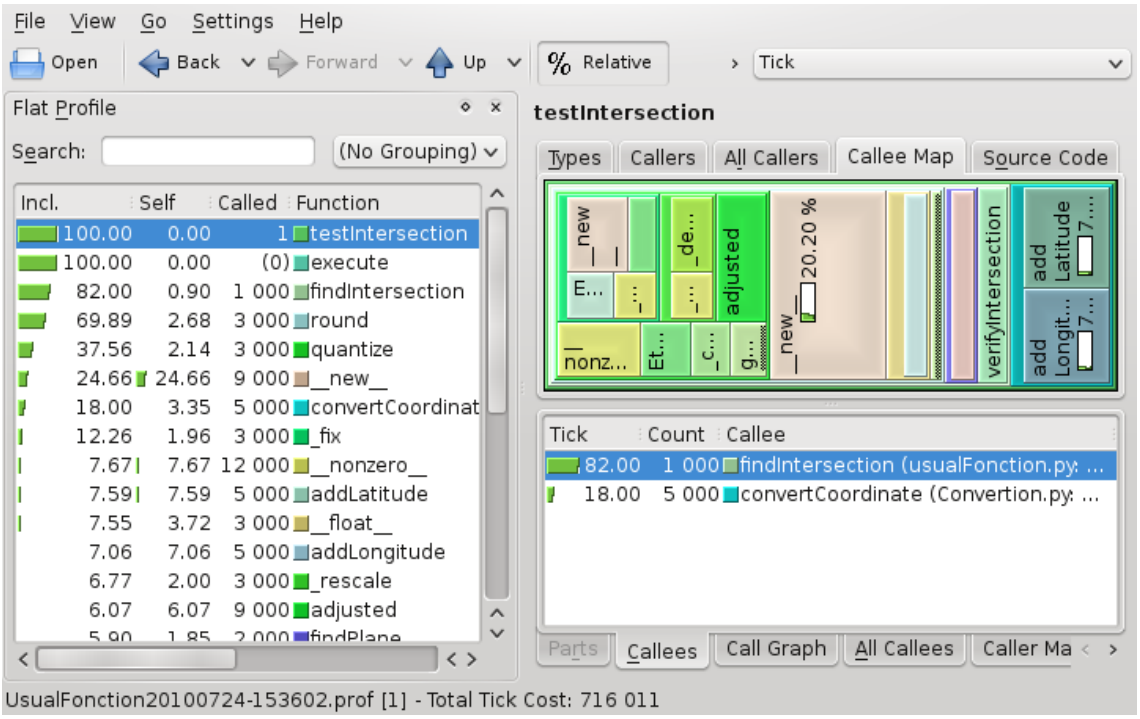


FIGURE **3.5** – Analyse avec Kcachegrind de l'exécution de la fonction "TestIntersection" du programme dans le but de l'améliorer.

# 4 Tests et validation de la réalisation

## 4.1 Les tests

Nous avons effectué au cours de ce projet deux types de tests : les tests unitaires et les tests globaux.

### 4.1.1 Les tests unitaires

En programmation informatique, le test unitaire est un procédé permettant de s'assurer du fonctionnement correct d'une partie déterminée d'un logiciel ou d'une portion d'un programme (appelée «unité» ou «module»).

On écrit un test pour confronter une réalisation à sa spécification. Le test définit un critère d'arrêt (état ou sorties à l'issue de l'exécution) et permet de statuer sur le succès ou sur l'échec d'une vérification. Grâce à la spécification, on est en mesure de faire correspondre un état d'entrée donné à un résultat ou à une sortie. Le test permet de vérifier que la relation d'entrée - sortie donnée par la spécification est bel et bien réalisée.

Petit rappel de définitions :

**Test :** il s'agit d'une vérification par exécution.

**Vérification :** ce terme est utilisé dans le sens de contrôle d'une partie du logiciel. (Une «unité» peut ici être vue comme «le plus petit élément de spécification à vérifier»)

Il s'agit pour nous de tester un module, indépendamment du reste du programme, ceci afin de s'assurer qu'il répond aux spécifications fonctionnelles et qu'il fonctionne correctement en toutes circonstances.

Mais ces tests ne sufisent pas car il ne donnent pas assez de recule pour visualiser si l'ensemble du programme est fonctionnel. Il nous donne seulement une confirmation théorique.

### 4.1.2 Les tests globaux

Comme nous l'avons cité précédement des test unitaires ne sont pas sufisant dans notre cas. En effet se projet etant un réél prototype dans le sens ou rien n'a été effectué de semblable auparavant, les spécification reste parfoit mal déterminée. On pourra cité comme exemple la scusture des message FPL qui sont cencé avoir toujours

la meme forme, mais qui se retrouve souvant avec des erreur du à la qualité de transmission du message.

Ces tests aurons donc pour but de valider le fait que toutes les parties développées indépendamment fonctionnent bien ensemble de façon cohérente.

Pour ce faire nous avons passé un grands nombre de fichier source à «parsé» Ce qui nous a permis de découvrire tout au long du projet un certains nombre d'erreur comme le fait de prendre en compte le nom de l'avion, aéroport de départ et heure de depart comme identifiant, celui-ci pouvant être le meme sur plusieur jours pour les vols cylciques. Dans ce cas les tests globaux nous on permis de decouvrir l'errur et nous a permis d'ajouter le jour de depart de l'avion dans l'identifiant afin que chaque identifiant reste bien unique.

## 4.2 La validation

Chaque fin de cycle de notre méthode de gestion de projet qui est l'Extreme Programing nous amene a une étape de validation. Celle ci conciste à verifier avec le client que le programme se comporte bien comme il le souhaitait.

Après chaque validation des spécification sont modifié car, bien que répondant à leur definition, elle ne réponde pas réelement au attente du client. D'autres spécifications sont crée et certaines annulées.

## 4.3 Amélioration continue

A partir des tests réalisés et de la validation avec le client comme cité précedement nous redéfinissons les besoins ainsi que les spécification. Cela nous amène donc à revenir au cycle des besoins (cf. 2.1 page 11) dans notre méthode de gestion de projet (cf. 2.2.2 page 13).

Nous reprenons alors un cycle ce qui nous permetra d'amélioré le programme en continu.

# 5 Synthèse

## 5.1 Spécifications obtenues

Comme vous l'avez compris, utiliser une méthode agile comme l'extreme programming (cf. 2.2.2 page 13) implique une perpétuelle réécriture des spécifications. Celle ci sont améliorée, réécrite, ajoutée tout au long du projet. C'est pourquoi dans ce rapport serra cité la dernière version des spécifications.

Les spécifications du logiciel sont les suivantes :

**Capture de fichiers de configuration :** Les points caractéristiques, route, zone de contrôle et ACI, doivent être récupérés dans les fichiers de configuration du système TIARE afin d'avoir la représentation la plus juste de ce que le système a. Ils doivent être gardé en mémoire pendent toute l'exécution du logiciel afin de pouvoir être utilisé. Les données seront enregistrées dans des objet le temps de l'exécution du programme afin de faciliter leur exploitation. La configuration du logiciel doit laisser à l'utilisateur la possibilité de spécifier le chemin du fichier de configuration.

**Capture des données Plan de vol :** Les données Plan de vol doivent être récupérées dans les log du système TIARE. Par contre il doit être possible de les récupérer d'un autre fichier contenant des trames FPL au format normalisé par la norme 4444 (cf Bib. [?]). Chaque plan de vol serra enregistré dans un objet ayant un identifiant comprenant : L'identifiant de l'avion, son aéroport de départ ainsi que l'heure et le jour de départ. Cet identifiant a pour but de les différentier et de les référencer dans le temps. La configuration du logiciel doit laisser à l'utilisateur la possibilité de spécifier le chemin du fichier de log.

**Capture des données ADS :** Les données ADS doivent être récupérées dans les log du système TIARE. Il devra être aussi récupérer dans ces log les points de la position en fonction du temps calculé par le système entre deux report ADS. Les reports ADS et points calculé seront instancié par avion et par vol. L'identifiant de chaque vol sera donc composé de l'identifient de l'avion ainsi que de la date et l'heure du message de login. La configuration du logiciel doit laisser à l'utilisateur la possibilité de spécifier le chemin du fichier de log.

**Les points caractéristiques :** Ces points devront être implémentés dans GOOGLE EARTH avec la possibilité de les afficher ou non. La configuration du logiciel doit laisser à l'utilisateur la possibilité de spécifier la possibilité de rééditer ou non le fichier source GOOGLE EARTH. Ces points serons représenté par un triangle de petite taille.

**Les zones de Contrôle et ACI :** Les zones de contrôle et zones ACI devront être

implémenté dans GOOGLE EARTH avec la possibilité de les afficher ou non. La configuration du logiciel doit laisser à l'utilisateur la possibilité de spécifier la possibilité de rééditer ou non le fichier source GOOGLE EARTH. Ces zones seront représentées par une surface colorée en 2 dimension.

**Les routes :** Les routes devront être implémentées dans GOOGLE EARTH avec la possibilité de les afficher ou non. La configuration du logiciel doit laisser à l'utilisateur la possibilité de spécifier la possibilité de rééditer ou non le fichier source GOOGLE EARTH. C'est route seront représentée par une ligne de couleur Jaune. Les points définissant cette route ne seront pas illustrés afin de ne pas faire de doublon avec les points caractéristique. Les coordonnées des points de chaque route devront être défini à partir des points caractéristique en mémoire.

**Les plan de vol :** Les plans de vol devront être implémenté dans GOOGLE EARTH avec la possibilité de les afficher ou non. La configuration du logiciel doit laisser à l'utilisateur la possibilité de spécifier la possibilité de rééditer ou non le fichier source GOOGLE EARTH. Les plans de vol doivent pouvoir être visualisé dans GOOGLE EARTH en fonction du temps. Pour se faire une heure théorique de passage sera calculé par le programme pour chaque point définissant le plan de vol. Toutes les informations concernant chaque plan de vol tel que ça route, les points constituant sa route et sa situation dans le temps devront être regroupé dans un dossier. Le message FPL de l'avion doit être visible dans la description de ce dossier. Les plans de vol seront visible durant toute la durée du vol et représenté par une ligne noire. La visualisation dans le temps sera représentée par un segment de couleur choisie aléatoirement pour chaque vol défini par les deux points les plus proches de l'heure en paramètre dans le logiciel (un point avant et un point après). Ce segment et ses points ne serons visible qu'à partir de l'heure du premier points jusqu'à l'heure du deuxième.

**l'intersection du plan de vol avec la zone ACI :** L'intersection, si elle a lieu, entre le plan de vol et la zone ACI doit être calculé, définie et représenté dans GOOGLE EARTH par un point rouge accompagné du nom de l'avion et de l'heure d'intersection affiché en rouge également. Ces points devront être contenu dans le dossier du concerné.

**Les report ADS :** Chaque report ADS serra composé de ces points de report ainsi que des points calculé par le système. Chaque report serra regroupé dans un dossier par vol et aura comme description le message reçu. Chaque point calculé par le système sera attribué et regroupé avec le report précèdent. Les vols seront représenté par une ligne blanche retraçant tout les report reçu, ainsi que chaque point affiché dans le temps. L'intérêt étant de visualisé l'écart entre le chemin parcouru par l'avion et le plan de vol déposé ainsi que la différence entre la trajectoire de l'avion et celle calculée par le système TIARE.

## 5.2 Gestion de projet

L'utilisation de l'extreme programming pour géré le projet aura été réellement bénéfique. On notera tout de même que cette méthode requière des clients extrêmement compétents et réactifs. En effet sans compétence de la part du client le projet peu rapidement tourner en rond.

Le fait de renouveler sans cesse les besoins et spécifications permet de réaliser un produit riche, adapté et performent. L'évolution quant a elle demande une maîtrise bien plus stricte qu'avec des méthode de gestion de projet plus classique sous peine de devenir rapidement désordonnée.

## 5.3 Le projet

Nous avons atteint un grand nombre d'objectif avec ce projet, il nous est capable d'analyser des plans de vol et de les mettre en corrélation avec les report reçu de l'avion par l'intermédiaire de liaison satellite.

Le projet n'est pas fini. Un grand nombre d'améliorations restent à implémenter tels qu'une interface graphique ou encore une base de donnée (cf. 6 page 37).

## 5.4 Un stage formateur

### 5.4.1 Un apport technique

J'ai pu au cours de se stage approfondir et mettre en pratique mes connaisance en programation python. Mais j'ai surtout pu découvrir le monde de l'aéronautique.

En effet ce stage ma permis de voir les technologies utilisées dans les zones de contrôles. J'ai donc pu prendre connaissance des technologies de détection des avions de dernière génération (des fois pas encore mis en place) tels que les radars secondaires Mode S. Ou encore leur système de détection de collisions. J'ai également pu m'instruire sur leurs solutions de télécommunication (Satellite, VHF) qui ne sont pas enseigné sous cet angles (pratique et non théorique) dans le cadre de mon cursus.

J'ai aussi pu découvrir le fonctionnement d'un serveur NTP, les principe de la paravirtualisation ou encore la mise en place de réseaux privé virtuel. Autant de domaines n'étant pas en relation direct avec le stage mais qui auront une grande utilité dans mon avenir professionnel.

Un autre point découvert dans la pratique aura été les méthode de gestion de projet. La première étant le cycle en V du fait que ce soit celle qui est appliqué au sein de l'entreprise. La deuxième étant l'extreme programing utilisée pour le projet du stage.

L'ouverture d'esprit du personnel faisant partie ou travaillant pour la DGAC y a fortement contribué.

### 5.4.2 Des rapport humains

Ce stage m'a également permis, notamment lors des pause café ou repas du midi, d'échanger avec un grand nombre d'ingénieur travaillant pour la DGAC ou faisant parti de l'entreprise.

Ce sont grâce à ces échanges que j'ai pu acquérir une grande partie des connaissances en aéronautique et en informatique citées précédemment. En effet ces personnes n'on pas héritées à prendre un peu de leur temps pour me faire des schémas sur le fonctionnement des différentes technologies de radars ou encore retrouver leur rapport de test (benchmark) réalisé sur différentes mise en place de virtualisations d'OS (Xen, Kve, VMWere).

Tout ces échanges auront été bénéfique, que se soit sur un plan purement technique, ou sur un plan relationnel.

## 5.5 Conclusion

Ce stage aura été une expérience professionnel très enrichissante. Il m'aura permis de découvrir le monde de l'aéronautique. Il m'aura aussi permis de découvrir plusieurs méthode de gestion de projet ainsi que des connaissances techniques variées.

Au delà des aspect pédagogique technique, il m'aura aussi permis de me familiariser avec le monde de l'entreprise du point de vue d'un ingénieur.

Le contexte m'aura aussi sensibilisé sur la qualité. En effet dans l'aéronautique la gestion de la qualité est des plus importante du fait qu'un système ne peut pas tomber en panne sous peine d'avoir de grave conséquences.

# 6 Evolution projet

Ce projet est loin d'être arrivé a termes. Nous allons donc voir ici ce qui pourrait être fait afin de perfectionner ce logiciel. Les évolution seront axées sur trois points :
- La mise en place d'une interface graphique.
- L'automatisation de l'acquisition.
- La pérennisation des données.

## 6.1 La mise en place d'une interface graphique

Comme il a été expliqué précédemment (cf. 3.4.1 page 22), la configuration du logiciel est effectuer manuellement par l'intermédiaire de fichiers textes et son exécution est effectuée en ligne de commande. C'est pourquoi une interface graphique faciliterait grandement son utilisation.

Cette interface devrait pouvoir faciliter la configuration et l'exécution du programme, elle pourrait être basée sur des technologie web afin de la rendre portable tout en séparent le traitement des données de l'utilisation du fichier final dans GOOGLE EARTH. En effet le programme pourrait être lancé a distance sur une machine, cela permettrait de sécuriser l'accès au données tout en libérant les ressources du poste de l'utilisateur.

Pour faciliter la configuration un histogramme avec tous les vol figurant entre deux date sélectionnée pourrait être réalisé, cela permettrait de mieux visualiser le trafic et de pouvoir cibler les vols a afficher.

Il pourrait aussi être intéressant d'inclure l'affichage final dans l'interface web, tout en laissant la possibilité a l'utilisateur de télécharger le fichier afin d'exploiter pleinement toutes les fonctionnalité du logiciel GOOGLE EARTH tel que la mesure de distance entre deux points.

## 6.2 L'automatisation de l'acquisition

Actuellement chaque fichier à traiter est récupéré manuellement. On pourrait concevoir un système qui irait de lui même chercher les fichiers nécessaire dans le système TIARE et les mettre automatiquement à la disposition du programme.

## 6.3 La pérennisation des données

Dans une optique de pouvoir rejouer simplement des situations passées, on pourrait mettre en place un système de base de données légère tel que SQLite[1]. Contrairement aux serveurs de bases de données traditionnels, comme MySQL ou PostgreSQL, sa particularité est de ne pas reproduire le schéma habituel client-serveur mais d'être directement intégrée aux programmes. L'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant de la plate-forme.

Ce procédé couplé à un traitement automatique permettrait de mettre et garder en mémoire tout les vols disponibles sur le système TIARE. Il permettrais donc de pouvoir rejoué des situations qui n'on été enregistrées plusieurs mois avant.

1. SQLite est une bibliothèque écrite en C qui propose un moteur de base de données relationnelles accessible par le langage SQL.

# Bibliographie

[1] KERVIZIC Emmanuel and internaute. Titre du thread. http://maths-forum.com/showthread.php?p=692707#post692707, june 2010.

[2] Patrick Fuchs and Pierre Poulain. Expressions régulières et parsing. http://www.dsimb.inserm.fr/~fuchs/python/python-node13.html, june 2010.

[3] Google. Documentation en ligne sur la sémentique des documents kml. http://code.google.com/apis/kml/documentation/kmlreference.html, june 2010.

[4] Python v2.7. Documentation en ligne de python. http://docs.python.org/, june 2010.

[5] Tareck Ziadé. *Programmation Python, Conception et optimisation*. Eyrolles, 2009.