

Raport de stage Eleve Ingenieur

Emmanuel KERVIZIC

10 septembre 2010

Table des matières

1	Introduction	3
2	Contexte	4
2.1	sujet du stage	4
2.2	Présentation de l'environnement	4
2.2.1	DSNA/DTI	4
2.2.2	Situation géographique	5
2.3	Contrôle aérien à Tahiti	5
2.3.1	Le système TIARE	5
2.3.2	La zone ACI :	6
3	Expression du besoin	7
3.1	L'objectif initial du projet	7
3.2	Les besoins	7
3.3	Les risques	8
4	Gestion de projet	9
4.1	Choix de la methode de gestion de projet	9
4.2	L' Extreme Programming	9
4.3	Le cycle en V	11
4.4	Les avantages et inconvenients	12
5	Rédaction des spécifications	13
6	Réalisation technique	16
6.1	Le contexte technique opérationnel	16
6.1.1	EUROCATX	16
6.1.2	Le domaine de l'aviation	17
6.2	Base de travail	18
6.2.1	Le langage Python	18

6.2.2	GOOGLE EARTH	19
6.3	Le programme réalisé et ses fonctions	19
6.3.1	Le fonctionnement	19
6.4	Problèmes techniques rencontrés et solution apportées	20
6.4.1	Gestion des erreur	20
6.4.2	Intersection entre plans de vol et zone ACI	21
6.4.3	Performance du logiciel	24
7	Tests et validation de la réalisation	25
8	Synthèse	26
9	Evolution projet	27

Chapitre 1

Introduction

Moi, le stage

Chapitre 2

Contexte

2.1 sujet du stage

Copier collé du sujet

2.2 Présentation de l'environnement

2.2.1 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 pôles qui 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.

Les partenaires

Tissu local, sous-traitant.

Relations au plan local, national...

2.2.2 Situation géographique

Son emplacement.

2.3 Le système de contrôle aérien mis en place à Tahiti

2.3.1 Le système TIARE

TIARE est un système de gestion du trafic aérien pour le centre de contrôle de Tahiti, en remplacement des systèmes vieillissants de visualisation du trafic (VIVO) et de gestion de plans de vol et d'informations générales (SIGMA). La superficie de

l'espace aérien géré par le centre de contrôle de Tahiti s'étend sur 12 500 000 km². 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.

2.3.2 La zone ACI :

Une fonction de contrôle spécifique, nommée ACI¹ 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 FIR² de Tahiti, dont 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. ACI : Area Common Interest, soit une zone d'intérêts commun

2. la FIR est la zone dans laquelle les contrôleurs doivent assurer le contrôle des vols

Chapitre 3

Expression du besoin

3.1 L'objectif initial du projet

L'objectif principal est 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 calcul 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.

3.2 Les 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 globale de ce qu'il souhaite en sortie. A savoir :

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

Par contre le besoins précis de l'utilisation du produit reste indéterminée. C'est pourquoi nous avons orienté notre gestion de projet vers une méthode dite agile (cf. [4.2 page 9](#)). 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 à 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 ...)

Tout au long du projet de nouveaux besoins sont apparus tel que :

- Représenter le trafic aérien en fonction du temps
- Définir approximativement l'heure d'entrée de et sortie des avions dans la FIR (cf. [2 page 6](#)) 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 estimés par le système TIARE entre deux reports ADS afin de visualiser l'interprétation des données reçues par le système.
- Différencier les types de vol en quatre catégories : Entrant, Sortant, Transit, Interne.

3.3 Les risques

Lorsque l'on a comme projet de réaliser une application qui a déjà été réalisée par le passé nous avons une base sur laquelle 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. [4.2 page suivante](#)) a été utilisée. Cette méthode nous a permis d'avancer petit à petit afin de susciter des besoins "réalisables". Contrairement à la méthode en V utilisée originellement à la DTI où les besoins et les spécifications sont déterminés avant le début de la réalisation technique.

Chapitre 4

Gestion de projet

4.1 Choix de la methode 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 claires afin de pouvoir réaliser un cycle en V (cf. [4.2 page 11](#)). Nous avons donc choisi 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 connaître réellement l'aboutissement final tout en gardant de la rigueur et de l'organisation. Nous avons donc décidé d'utiliser une méthode dite agile¹. Après quelque recherche et comparaison notre choix est orienté sur l'extreme programming (cf. [4.2](#)) nous allons donc vous décrire cette méthodologie et la comparer avec le système utilisé habituellement.

4.2 L' Extreme Programming

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".

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.

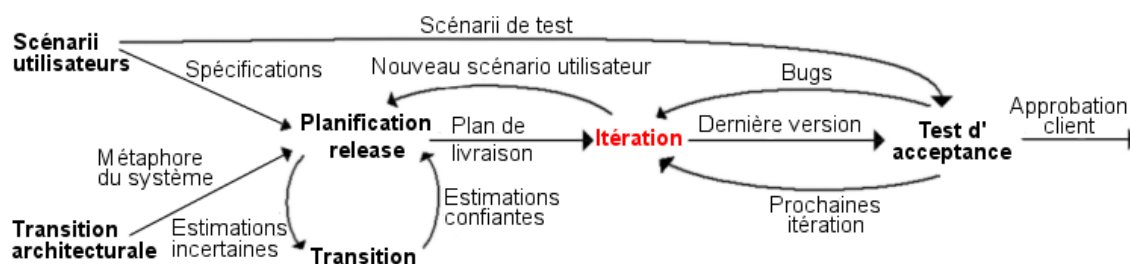


FIGURE 4.1 – Cycle de l'Extreme Programming.

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.

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. 4.1). 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)

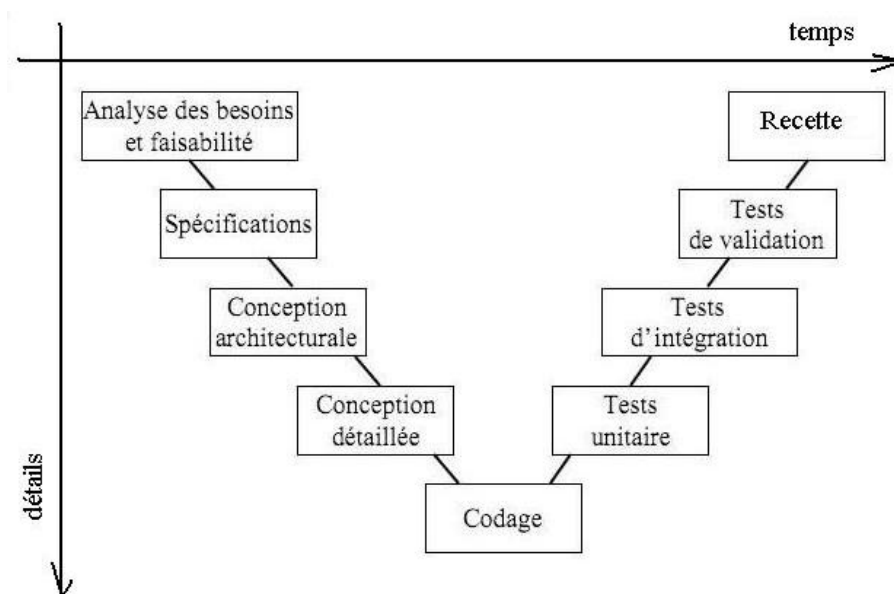


FIGURE 4.2 – Les phases à travers le temps et le niveau de détails.

4.3 Le cycle en V

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. 4.2) 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 constitue 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 indéfinis, il est impensable d'appliquer une telle méthodologie sans engendrer le risque que le logiciel final ne marche pas ou ne réponde pas aux attentes du client.

4.4 Les avantages et inconvénients

Les avantages de cette méthode, dans cette situation, sont :

- Enrichir le produit à chaque itération du cycle. Si le logiciel marche le client peut visualiser immédiatement les besoins qui étaient superficiels, dont il n'avait pas réellement besoin, 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 le meilleur délai (changement d'objectifs ou de priorités)
- ... à compléter.

Cette méthode implique tout de même un certain nombre d'inconvénients 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 si 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 Maître d'ouvrage qu'au niveau maître 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.
- ... à compléter.

Chapitre 5

Rédaction des spécifications

Comme vous l'avez compris, utiliser une méthode agile comme l'extreme programming (cf. ?? page ??) implique une perpétuelle réécriture des spécifications. Celle ci sont améliorée, réécrite, ajoutée tout au long du projet. C'est pourquoi dans ce rapport sera 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 gardés en mémoire pendant toute l'exécution du logiciel afin de pouvoir être utilisés. Les données seront enregistrées dans des objets 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 logs 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 sera 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 différencier 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 logs du système TIARE. Il devra être aussi récupérer dans ces logs les points de la position en fonction du temps calculé par le système entre deux reports ADS. Les reports ADS et points calculés seront instanciés par avion et par vol. L'identifiant de chaque vol sera donc composé de l'identifiant de l'avion ainsi

que de la date et l'heure du message de login. La configuration du logiciel doit laisser a l'utilisateur la possibilité de spécifier le chemin du fichier de log.

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

Les zones de Contrôle et ACI Les zones de controle et zones ACI devront être implémenté dans GOOGLE EARTH avec la possibilité de les afficher ou non. La configuration du logiciel doit laisser a l'utilisateur la possibilité de spécifier la possibilité de rééditer ou non le fichier source GOOGLE EARTH. Ces zones seront représentée par une surface colorée en 2 dimension.

Les routes Les routes devront être implémenté dans GOOGLE EARTH avec la possibilité de les afficher ou non. La configuration du logiciel doit laisser a l'utilisateur la possibilité de spécifier la possibilité de rééditer ou non le fichier source GOOGLE EARTH. Cest route seront représentée par une ligne de couleur Jaune. Les points definissant cette route ne seront pas illustré afin de ne pas faire de doublon avec les points caractéristique. Les coordonnée des points de chaque route devront être défini a 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 a l'utilisateur la possibilité de spécifier la possibilité de rééditer ou non le fichier source GOOGLE EARTH. Les plan 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 definissant le plan de vol. Toutes les informations concernant chaque plan de vol tel que ca route, les points constituant sa route et sa situation dans le temps devront etre 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 représenté par un segment de couleur choisie aléatoirement pour chaque vol défini par les deux points les plus proche de l'heure en parametre dans le logiciel(un point avant et un point après). Ce segment et ses points ne seront visible qu'a partir de l'heure du premier points jusqu'a l'heure du deuxieme.

l'intersection du plan de vol avec la zone ACI L'intersestion, 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 sera composé de ces points de report ainsi que des points calculé par le système. Chaque report sera 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 précédent. Les vols seront représentés par une ligne blanche retraçant tous les reports reçus, ainsi que chaque point affiché dans le temps. L'intérêt étant de visualiser 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.

Chapitre 6

Réalisation technique

Avant de passer à la pratique un apprentissage théorique à du être réalisé.

6.1 Le contexte technique opérationnel

6.1.1 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. [6.1 page suivante](#)), EUROCATX récupère les information sur les plans de vol par l'intermédiaire de CAGOU¹. Il récupère aussi le positionnement émis par l'avion à l'aide de la transmission Satellite, VHF² ou des données radars lors de son approche. Le système EUROCATX donne un accès à la bureautique protégé par un par-feu (Fire-Wall) 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 réseaux RSFTA.
- Tous les report ADS reçu par satellite et traité par le système.

Le système envoyé les information récoltées et celle calculées au visues³ situées dans la tour de contrôle au niveau de la Vigie ou de la salle CCR ainsi que de la position déportée à MOREA.

1. CAGOU : nom donné au commutateur RSFTA

2. VHF : Very High Frequency, soit une bande radio de très haute fréquence

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

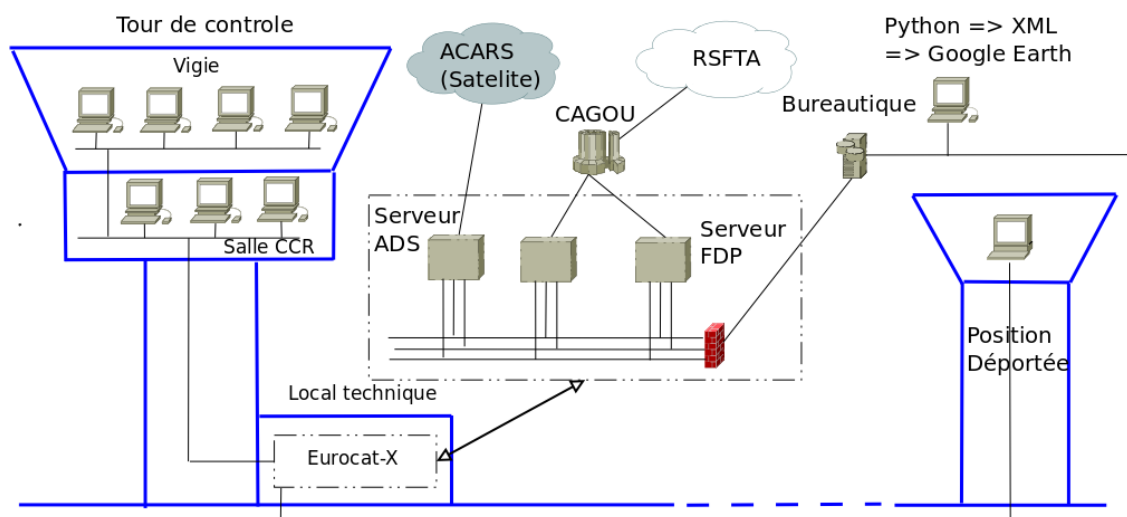


FIGURE 6.1 – Schématisation du système EUROCATX au niveau des tour de contrôle

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

6.1.2 Le domaine de l'aviation

Il m'a aussi été nécessaire de prendre connaissance de tous les termes, unités, conventions et j'en ai passé utilisés dans le domaine aéronautique.

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 expliqué ci dessus (cf. 6.4.2 page 21) me remémorer la manière de calculer le point d'intersection de deux arcs de cercle en coordonnées sphériques.

Convention :

Plusieurs conventions ont dû être acquises comme celle utilisée par le système TIARE pour décrire les report ADS ou encore celle utilisée par les compagnies pour le dépôt de plan de vol. NE PAS OUBLIER DE FAIRE RÉF AU DOCUMENT 4444 ...

6.2 Base de travail

6.2.1 Le langage Python

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 m'a permis 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:"`)

Utiliser les expression régulière :

L'apprentissage de l'utilisation des expression régulière⁴, 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.

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.

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.

Il c'est avéré à plusieurs reprises que certaines parties de mon code étaient très gourmandes en processus. L'apprentissage des fonction de test du code tel que le module hotshot décrit plus tard (cf. 6.4.3 page 24) m'a été rapidement nécessaire.

6.2.2 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 aériennes ou satellites. Ce logiciel donne la possibilité de configurer un environnement, ajouter des lignes, des points ou encore des polygone en 3D en passant par des fichier de configuration au format KML⁵.

Ce format, qui repose sur le XML⁶, a l'avantage d'être simple à manipuler. Ça sémantique est définie sur le de google (cf. Bibliographie [3])

6.3 Le programme réalisé et ses fonctions

6.3.1 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 fichier 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 différent 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 Shell⁷, par l'intermédiaire de

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

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 actions à effectuer.
2. Lire les fichiers de configuration du système TIARE afin de récupérer toutes les variables nécessaires sous forme d'objet⁸ (ex : points caractéristiques ...)
3. Lire les fichiers de log afin de créer des objets tels que les plans de vol ou encore les reports ADS. Ces objets sont créés non seulement à partir de ses fichiers de log mais aussi à partir des objets créés précédemment (ex : les points des plans de vol désignés par un nom sont convertis en coordonnées à l'aide des points caractéristiques).
4. Créer le fichier KML désigné dans le fichier de configuration à l'aide des objets instanciés.
5. Créer un fichier KMZ à l'aide de tous les fichiers KML afin d'avoir un fichier compact et facile à transporter.

6.4 Problèmes techniques rencontrés et solutions 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 dû faire face.

6.4.1 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 dû être gérées. Deux types d'erreurs sont revenues :

- Le premier type d'erreur était par exemple une réaction inattendue du logiciel, On pourrait prendre en exemple la conversion de coordonnées reçue en Système

8. Objet : structure de données évaluées et cachées qui répond à un ensemble de messages. Cette structure de données définit son état tandis que l'ensemble des messages qu'il comprend décrit son comportement

- sexagésimal⁹ en coordonnées utilisées dans les fichiers KML [5 page 19](#), qui lors des premiers test donnais des donnée erronées.
- Le deuxième type était celui du au erreur contenu dans les fichiers de log utilisé pour récupérer les informations. Ces erreur faisait 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 correspond 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 créés 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.

6.4.2 Intersection entre plans de vol et zone ACI

Problématique : Afin de déterminer l'heure d'entrée approximative des avions dans la zone ACI (cf. [2.3.2 page 6](#)) 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

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

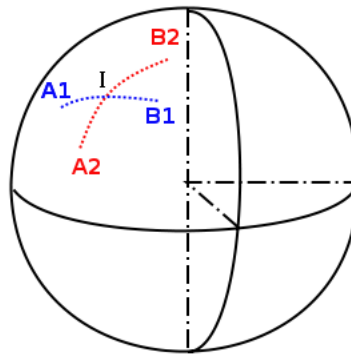


FIGURE 6.2 – Représentation grossière de l'intersection de deux arc de cercle respectivement formé par la trajectoire la plus courte entre deux points situé sur le Globe terrestre

cercles qui sont composé de deux extrémités définies par des points en coordonnées sphériques (cf. schéma fig. 6.2).

Résolution : Étant donné que j'ai effectué un BTS avant d'intégrer l'EIGSI¹⁰, les notion de coordonnées sphérique ne me sont que peut familière. Après avoir en vain 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 forcément 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.

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. 6.2). 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

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

– 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} :

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

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

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

6.4.3 Performance du logiciel

Problématique : Les premiers tests du logiciel se sont déroulés sur un nombre limité de fichiers (représenté par un nombre limité d'heure de vol), ce afin 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 mettait des heures avant de pouvoir sortir un fichier. Il a donc fallu optimiser le code afin d'alléger le programme en ressources.

Résolution : En cherchant des conseils dans des forums d'informatique ainsi que dans le livre cité précédemment (cf. bibliographie [5], nous avons découvert que Python était un langage orienté par les tests et qu'il disposait donc de bibliothèques spécialement conçues pour déterminer les points bloquants d'un programme et les fonctions appelées les plus gourmandes.

La fonction retenue pour repérer ce qui est appelé en anglais les Bottleneck¹¹ est la fonction "hotshot" qui a 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 à la figure ?? page ??.

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

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

Chapitre 7

Tests et validation de la réalisation

Tests et validation de la réalisation Démarche pour tester le produit (manque pas des vols) Un fichier même vol, mais fichiers avec des vols supplémentaires Présentation du rendu Améliorations continues : à partir des tests, je repars dans le chapitre précédent (réalisation technique + nouveaux besoins (comparaison FPL//ADSC) ou correction)

Chapitre 8

Synthèse

Synthèse Méthode employée à consommateur de personne à disposition, produit très riche si compétence, adapté et performant, Evolution désordonnée si pas maîtrisé (base de données en plus), changement des spécifications en cours de projet, difficulté de rédaction de spécification produit fini concentre sur le dev et moins sur la doc. Pas de rédaction de manuel d'utilisateur,

Chapitre 9

Evolution projet

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 sémantique 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.