Rapport

PIE Dassault Aviation : Assistance monopilote

Andrioaie Daniela - Angelier William - Garzon Joli Antoni - Lavisse Côme

Année académique 2020-2021

ISAE-SUPAERO

Table des matières

Introduction

Problématique générale

Le pilotage d'un avion, commercial, privé, civil ou militaire, demande une charge mentale élevée au pilote qui doit concentrer son attention sur différentes sources d'informations. Les jets civils de Dassault Aviation n'échappent pas à la règle. Il est donc imaginable de soulager l'équipage technique par l'intervention d'un assistant virtuel capable d'apporter les éléments demandés par le pilote à l'aide d'une intelligence artificielle. Dans la perspective de développement du "Single Pilot Operating (SPO)", notamment dans les jets d'affaires, cet outil serait particulièrement pertinent pour prendre en charge une partie du travail du copilote alors absent.

Un précédent PIE réalisé en 2019 a recensé les cas d'utilisation d'un tel système, c'est-à-dire les phases de vol à charge de travail élevée où l'équipage a besoin d'informations précises rapidement : approche, pannes, déroutement, etc. Tout ceci a été regroupé dans ce que nous allons appeler une ontologie. Elle peut être comparée une classification en arbre des données, regroupées par classe et type d'information. Avec les scenarii qui enrichissent cette ontologie, on obtient une modélisation complète de l'environnement direct de l'appareil. On peut alors effectuer des recherches de données en parcourant l'arbre pour chaque requête [1]. On dispose donc d'une ontologie regroupant les phases principales du déroulement d'un vol. En 2019, le projet avait abouti à une maquette codée en Python, permettant aux pilotes d'exprimer leur requête sous forme d'un texte écrit puis de récupérer la réponse également par écrit via une interface graphique. Cette première version d'IHM a servi de preuve de concept et a démontré l'utilité et la faisabilité du système. Cependant, son utilisation présentait de sérieuses limites, comme la formulation des requêtes par écrit qui monopolisait du temps et l'attention des pilotes ou encore la contrainte de formuler chaque requête avec une syntaxe très précise, solution peu ergonomique.

L'objectif principal de notre projet est d'améliorer le système existant afin que chaque requête puisse être formulée sans syntaxe précise, en langage naturel, de sorte que le pilote puisse communiquer avec l'assistant comme s'il parlait à son copilote. D'autres améliorations pourront ensuite être envisagées, comme l'ajout d'un assistant vocal (formulation des requettes à l'oral et réponse de l'IHM sous format audio), ou encore le développement ergonomique de l'interface.

Objectifs du projet

Notre objectif global est donc de se rapprocher au maximum d'un assistant virtuel utilisable en vol lors des phases à haute charge de travail. Ceci étant une tâche particulièrement complexe et longue, nous avons précisé et hiérarchisé nos objectifs en fonction des attendus de Dassault Aviation. Ceux-ci sont donc, par ordre de priorité :

1. Implémenter la compréhension de requêtes en langage naturel. Le système devra avoir été testé et fonctionner sur des requêtes types en anglais recueillies auprès de pilotes, dans le cadre de l'ontologie.
2. Ajouter la possibilité pour le pilote de poser ses requêtes à l'oral en anglais, avec un taux de compréhension par le système acceptable.
3. Compléter l'ontologie existante en prenant en compte des nouveaux scenarii recueillis dans la littérature et au travers d'entretiens avec des pilotes.
4. Améliorer, à travers une brève étude de neuroergonomie, l'interface utilisateur (entrées orales ou écrites, sorties orales ou visuelles, etc.).

L'objectif principal attendu par le client correspond au premier point. Cependant, face aux outils déjà existants, il nous est difficile de prédire quelle sera la difficulté pour les adapter à la reconnaissance du langage technique aéronautique. Nous avons donc enrichi les objectifs et construit le projet autour de l'ensemble, en étant prêts à adapter l'attribution des ressources selon les difficultés rencontrées.

Parties prenantes et attendus

Notre équipe de quatre étudiants, chargée de la réalisation du projet, sera accompagnée et encadrée par différentes personnes. Côté client, ce sont Hervé Girod et Timon Ther qui seront nos contacts chez Dassault Aviation. Après avoir défini et étudié ensemble leurs attentes, nous leur ferons part de nos avancées et les consulterons sur les principaux choix techniques. Leur attendu est en premier lieu une version fonctionnelle du système pouvant récupérer des requêtes en langage naturel. Les autres objectifs évoqués précédemment sont définis comme secondaires. Le client n'attend pas de livrable intermédiaire.

Pour la partie gestion de projet, nous serons accompagnés par Christel Cerclier. Les attendus sur ce point ont été définis à la première séance le 14 octobre, le premier étant ce livrable. Sur les aspects techniques, notre référente à l'ISAE-SUPAERO est Caroline Chanel, elle pourra nous aider notamment sur les aspects d'intelligence artificielle. Nous avons également identifié Ruffin VanRullen comme pouvant nous être d'un grand support, l'une de ses spécialités étant la reconnaissance du langage par Deep Learning.

Chapitre 1

Partie Management Projet

1.1 Equipe et méthode de travail

1.1.1 Présentation de l'équipe

Le projet que nous allons mener se divise en deux grandes parties : enrichissement des parties déjà élaborées lors du premier projet et implémentations de nouvelles fonctionalités. L'enrichissement concernera principalement l'ontologie. Pour le reste, nous concentrerons la majorité de nos ressources sur le passage de requêtes par mots clés aux requêtes en langage naturel.
Pour ce qui concerne la partie technique du projet, l'équipe se divise en deux. Deux personnes se concentreront sur l'amélioration de l'ontologie (déjà existante) et les deux autres sur l'ajout de nouvelles fonctionnalités.

- Côme Lavisse : Chef de projet et fonctionnalités ajoutées
- Antoni Garzon Joli : Responsable fonctionnalités ajoutées
- William Angelier : Fonctionnalités ajoutées et Amélioration du produit
- Daniela Andrioaie : Amélioration du produit

Bien que les restrictions sanitaires nous obligent à travailler majoritairement à distance les uns des autres, nous avons gardé un point hebdomadaire de suivi afin de s'assurer que les tâches de chacun restent disjointes, ou menées avec une bonne répartion des rôles. Côté client, nous avons effectué un point d'avancement mensuel en début de projet. En effet, le démarrage du projet étant essentiellement axé sur les aspects organisationnels, nous n'avions pas jugé utile d'en faire plus. Par la suite, lorsque la partie technique a commencé à se développer, nous avons réalisé des points hebdomadaires afin de montrer l'avancée du projet. Le but était également de s'entretenir avec le client afin de déterminer les objectifs réalistes de fin du projet. Comme nous avions proposé plusieurs améliorations de la maquette sans savoir le temps réel que cela prendrait, nous nous devions d'avoir une méthode avec des ré-ajustements itératifs.

1.1.2 Stratégie de travail

Nous avons fait le choix d'une méthode de gestion de projet traditionnelle. Le projet est divisé en tâches et sous-tâches qui sont interdépendantes et doivent donc être réalisées dans un ordre précis. Compte tenu de la grande incertitude sur les difficultés rencontrées dans les phases de programmation, il sera important d'actualiser régulièrement le plan de développement. Si cette incertitude aurait pu nous mener à choisir une méthode agile, nous avons préféré rester dans un cadre connu et se fixer de nombreux jalons pour pouvoir réagir rapidement en cas de retard.

1.1.3 Définition des jalons

Les jalons que nous avons respecté au cours du projet sont les suivants :

— 14 Novembre 2020 : Première définition des attendus clients ;
— 3 Décembre 2020 : Première revue côté gestion de projet ;
— 27 Janvier 2021 : Deuxième revue côté gestion de projet ;
— 16 Février 2021 : Re-définition des attendus clients ;
— 24 Mars 2021 : Remise du rapport ;
— 31 Mars 2021 : Soutenance.

Les revues de gestion de projet sont imposées par l'école. Pour la définition et la re-définition des attentes, nous avons choisi ces dates qui étaient les plus judicieuses selon nous, compte tenu du décallage entre les développement gestion et technique du projet.

1.2 Décomposition du produit et des activités

1.2.1 PBS et WBS

Concernant les spécifications du produit, on retrouve le PBS en figure 1.1.

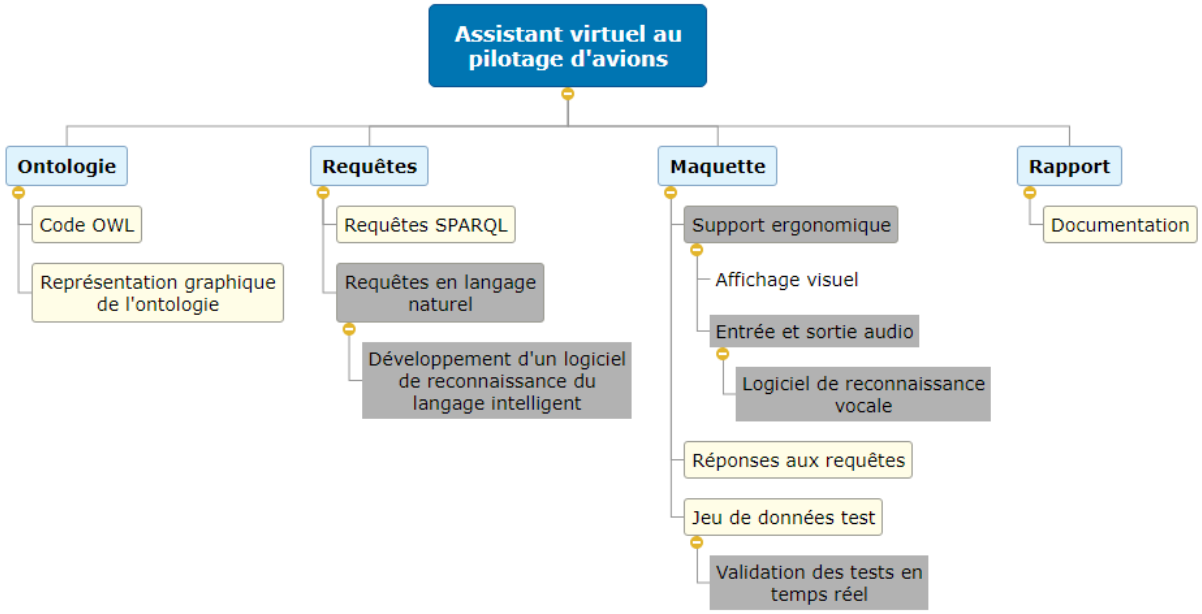

Fig. 1.1 – PBS

Les cases qui sont grisées correspondent aux spécifications du produit qui seront ajoutées par rapport au projet de l'année dernière. Les aspects du produit en blanc seront pour certains améliorés, comme par exemple l'ontologie qui sera complétée et l'ergonomie de la maquette qui fera fera l'objet d'une étude plus poussée sur les facteurs humains.

La décomposition des activités (WBS en figure 1.2) découle de cette décomposition du produit mais ne concerne que les actions qui seront réalisées par notre groupe. Les éléments obtenus sont décomposés et parfois regroupés pour en faire les tâches qui apparaîtront dans notre planning.

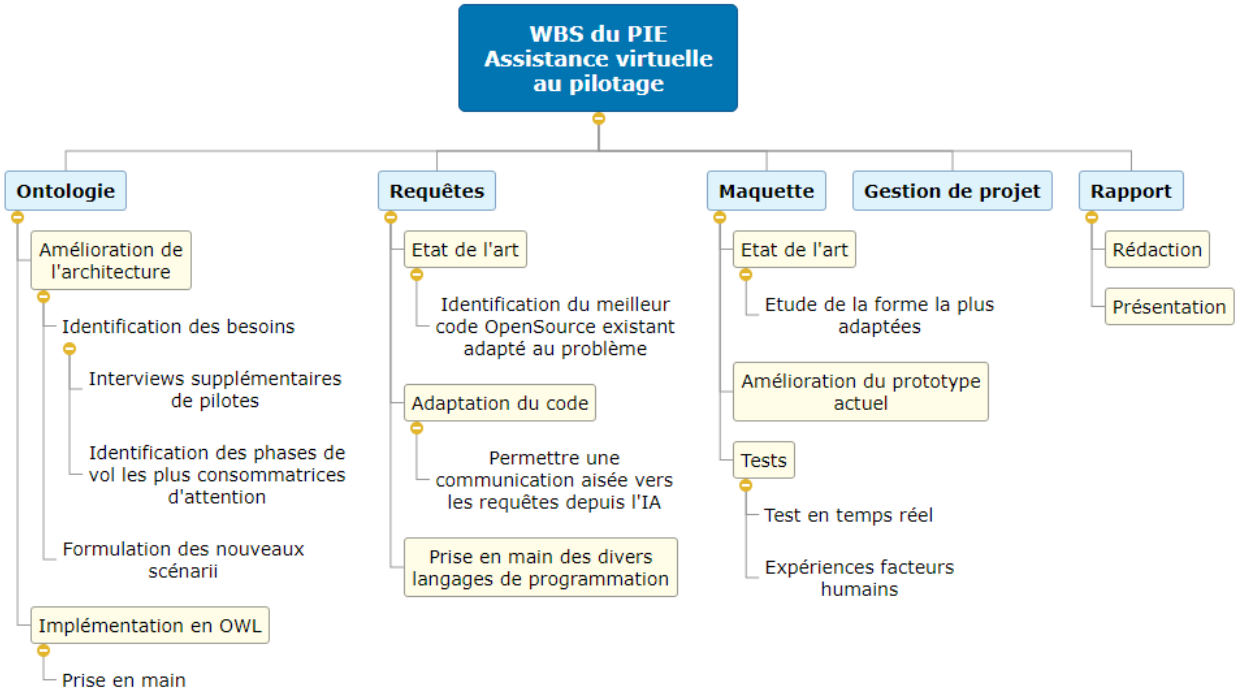

Fig. 1.2 – WBS

1.2.2 Organisation de l'équipe

A partir de cette décomposition, nous avons établi une matrice des responsabilités visible en figure 1.3. Compte tenu de l'organsation en objectifs principaux et secondaires du projet, chaque membre du groupe interviendra sur les tâches principales et sera affecté à une tâche annexe. Nous avons choisi un responsable (accountable) par sous-tâche chargé de surveiller l'avancement de celle-ci et de remonter les problèmes le cas échéant. Celui-ci participe égalment à l'avancée de la tâche (Responsible, non indiqué sur la matrice). Pour la tâche "Soutenance PIE", nous sommes tous acteurs des livrables. Nous avons décidé de ne pas nommer un responsable (A) en particulier afin que cette responsabilité soit répartie sur l'équipe projet dans sa globalité.

1.3 Organisation temporelle du projet

1.3.1 Planning

Pour établir le planning du projet, il a été indispensable de définir les relations de précédence entre les tâches. Globalement, le développement de l'ontologie est indépendant du reste. La réalisation de notre objectif principal est au contraire indispensable avant de débuter les suivants. Avant d'envisager une reconnaissance vocale des requêtes, il faut un assitant virtuel qui comprenne efficacement le langage naturel. De même, avant de commencer la campagne d'expériementation et de validation auprès des pilotes, il est indispensable que la maquette soit fonctionnelle. Elle pourra ensuite être améliorée.

Ces relations entre les tâches sont représentées sur le diagramme de Gant en annexe (5.4). On peut observer le chemin critique qui part de la compréhension du besoin client, passe par l'implémentation de la reconnaissance du langage ouvert, par le développement de la reconnaissance vocale puis finit par la phase de tests et d'améliorations. En cas de retard important, ce chemin peut être adapté et

Tâches	Sous-tâches	Daniela ANDRIOAIE	William ANGELIER	Antoni GARZON JOLI	Côme LAVISSE	Hervé GIROD	Timon THER
Gestion de projet	WBS, OBS, Planning, Risques, Opportunités	R	R	R	A	I	I
	Rédaction du plan de développement	R	R	R	A	I	I
Projet	Enrichissement de l'ontologie	C	A	R	C	C	C
	Etat de l'art sur la reconnaissance du langage	C	C	R	A	C	C
	Adaptation du code et intégration	R	R	R	A	C	C
	Mise en place de la reconnaissance vocale	A	R	C	C	C	C
	Tests et expérience facteurs humains	C	R	A	R	C	C
Soutenance PIE	Revue technique	R	R	R	R	I	I
	Revue de projet	R	R	R	R	C	C
	Livraison des fichiers techniques	R	R	R	R	I	I

R	Responsible
A	Accountable
C	Consulted
I	Informed

Fig. 1.3 – Matrice RACI

raccourci en abandonnant certains objectifs secondaires. La définition des jalons permet de placer des points de contrôle dans notre planning. Lors des deux revues de projet en janvier puis février, le plan de développement sera mis à jour. Le plan de charge sera complété avec le budget effectif passé, et l'analyse des risques sera refaite en fonction des contre-mesures mises en place.

En réalité, le différentes tâches du projet ont pu être menées de manière plus indépendantes que prévu. La programmation de la compréhension du langage naturel a été sous-estimée, mais en utilisant la méthode simpliste mise en place par l'équipe 2019 nous avons pu tester des solutions de reconnaissance vocale avant la fin du projet. Ceci a permis une meilleure répartition des rôles, et donc un gain d'efficacité conséquent dans la gestion du temps. Le Gantt représentant l'organisation réelle du projet une fois mené à sont terme est présenté en annexe (1).

1.3.2 Budget prévisionnel

Le projet ingéniérie entreprise à l'ISAE-SUPAERO compte pour 5 crédit dans notre programme annuel. Cela constitue officiellement 120 heures par personne impliquée dans le projet soit 480 heures de travail au total. Sur l'emploi du temps, 80 heures sont alouées par personne au projet soit 320 heures. Ces deux nombres nous donnent une fourchette réaliste de la quantité de travail à fournir. La répartition des heures dans les différentes tâches de ce projet est détaillée dans la table 1.1. Ce tableau suit les tâches décrites dans le diagramme de Gantt illustré en annexe. Nous avions effectué un budget prévisionnel au début du projet et il a été révisé mi-décembre. Nous avons ensuite mis le nombre d'heures que nous avons effectivement passé sur chacun des sujets. Nous avons utilisé ce tableau au cours de notre projet pour ne pas mettre trop de ressource sur l'un ou l'autre sujet. Sans ça, nous aurions pu passer uune centaine d'heure sans s'en rendre compte sur un problème en oubliant le temps que l'on s'était fixé.

On remarque également que sur ce budget, il y a de nouvelles tâches qui sont apparues en cours de projet. En effet, en fonction de notre avancée, des attendus clients et de ce que nous pensions être réalisable pour mars 2021, nous avons ajouté ou supprimé des tâches initialement prévues.

Tâches	Prédiction Budget	Budget réel
Partie Technique		
Compréhension des attentes client	16h	20h
Enrichissement de l'ontologie	30h	22h
Etat de l'art	0h	10h
Reconnaissance du langage ouvert		
Recherches bibliographiques	12h	25h
Selection de l'architecture	8h	2h
Prise en main de la solution	20h	10h
Intégration au code	40h	31h
Tests sur requêtes écrites	10h	10h
Reconnaissance vocale des requêtes		
Recherches bibliographiques	12h	10h
Prise en main de la solution	15h	26h
Intégration au code	20h	5h
Tests et améliorations		
Elaboration des tests	20h	5h
Tests en interne	15h	10h
Amélioration de la maquette	40h	66h
Tests performance	0h	8h
Tests facteurs humains avec pilotes	10h	2h
Sous total	**268h**	**262h**
Partie Management		
Rédaction du plan de développement	16h	15h
Mise à jour du plan de développement	10h	5h
Rédaction du rapport	40h	55h
Préparation de la soutenance	10h	à venir
Sous Total	**76h**	**75h évolutif**
Total	**344h**	**337h**

TABLE 1.1 – Budget prévisionnel et réel en heure

1.3.3 Plan de charge

A partir du planning complet du projet et de la charge de travail associée à chaque tâche, nous sommes en mesure de représenter le plan de charge jusqu'à la livraison. Le Gantt et ce plan ont été réalisés ensemble de manière itérative afin de lisser au mieux la charge sur la durée du projet, en prenant en compte l'avancée au moment de la rédaction de ce document. La plus grosse charge sera répartie sur les mois de décembre, janvier et février avec un peu moins d'heures placées en décembre pour prendre en compte les vacances. Les heures réalisées seront suivies et régulièrement comparées à ce plan.

Le point intermédiaire de janvier nous a fait prendre conscience du retard accumulé en décembre. Les objectis horaires ont donc été revus à la hausse afin d'atteindre les objectifs avant le jalon final. Cette méthode a été efficace et nous a permis de garder des objectifs réalistes tout au long du projet.

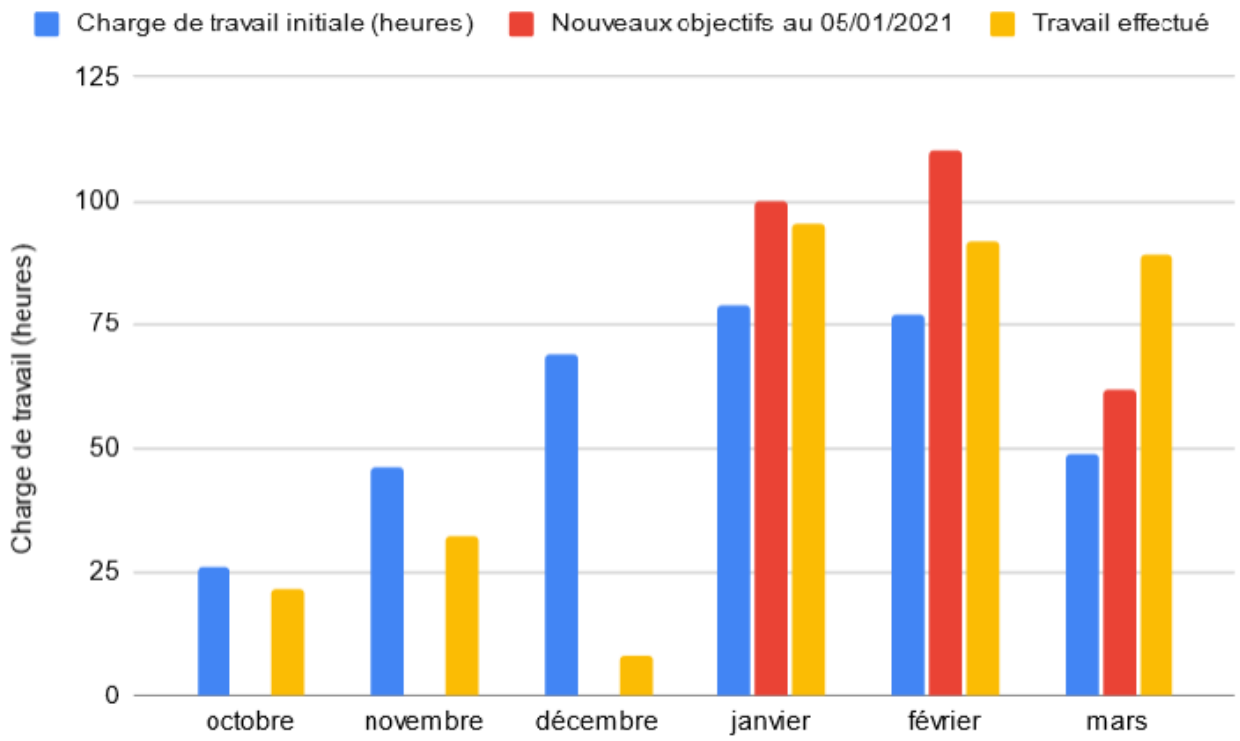

Fig. 1.4 – Plan de charge

1.4 Suivi et contrôle

1.4.1 Suivi des activités

La situation particulière de la crise sanitaire et à fortiori du confinement nous imposent une rigueur particulière sur tous les échanges au sein du groupe et avec les parties prenantes. Nous nous attendons à poursuivre le projet en distanciel et adaptons en conséquence nos outils de communication. Les réunions au sein du groupe se feront sur Zoom une fois par semaine, avec des échanges fréquents sur une application de messagerie. Les échanges avec le client se feront par Skype, une à deux fois par mois selon les questions à poser et les décisions à prendre ensemble. Les tuteurs sont particulièrement réactifs et enclin à nous répondre en cas de question non programmée sous forme de réunion.

Techniquement, notre PIE ne nécessite pas de matériel complexe si ce n'est un ordinateur pouvant éxécuter un code Python et un environnement de travail adapté aux langages utilisés. Les documents sont partagés au sein du groupe sur un drive, et nous utilisons Git pour partager nos avancées sur les phases de programmation et gérer le versionnage de notre produit. Les tests réalisés auprès des pilotes n'ont pas nécessité de matériel supplémentaire : un test plus poussé sur simulateur a été envisagé mais n'a pas été réalisé du fait de la situation sanitaire.

Un document partagé a permis à chacun d'indiquer le temps passé sur le projet, à quel moment et sur quel sujet. Le chef de projet complètera en parallèle le Gantt en remplissant l'avancée des tâches, sur l'outil GanttProject. Le plan de charge et le planning ont donc été actualisés régulièrement en fonction de l'avancement du projet.

1.4.2 Maîtrise des risques

Afin d'anticiper tout évènement pouvant compromettre la réalisation du projet dans le temps imparti, nous avons réalisé une analyse des risques encourus. Nous avons répertorié 14 risques répartis dans 4 catégories de gravité par rapport à leur probabilité occurrence et leur impact.

Num.	Catégorie	Description			Evaluation pré-risque	
		Causes	Evènement à risque	Conséquence	Probabilité	Impact
1	client	Licenciement, conflit, décision de la hiérarchie	Perte de contact avec le client	Arrêt du projet	Faible	Grave
2	client	Redéfinition tardive des objectifs	Refonte totale du projet	Les livrables ne sont plus en adéquation avec l'objectif	Faible	Grave
3	humain	Conflit personnel au sein du groupe	Perte de cohésion	Perte de temps à gérer le conflit	Faible	Grave
4	humain	Démotivation d'un membre du groupe	Un membre ne travaille plus	Perte de capacité de travail et de temps	Faible	Moyen
5	organisation	Mauvaise communication entre les trois équipes	Divergence des architectures entre les trois équipes	Prise de retard sur les déliverables	Fort	Moyen
6	organisation	Problèmes de communication dans l'équipe dues au télé-travail	Mauvaise coordination des tâches à effectuer	Prise de retard sur les déliverables	Moyenne	Moyen
7	organisation	Mauvaise estimation du temps prévu pour une tâche	Prise de retard sur une tâche	Prise de retard sur les déliverables si la tâche est située sur un chemin critique	Moyenne	Moyen
8	technique	Manque de compétences en OWL	Impossibilité d'implémenter le code	Impossibilité d'implémenter l'ontologie	Moyenne	Grave
9	technique	Manque d'information, hypothèses fausses	Mauvais choix d'architecture	Modélisation inutilisable	Moyenne	Grave
10	technique	Problème d'implémentation	La maquette ne fonctionne pas	Impossible de valider le fonctionnement de l'ontologie	Faible	Grave
11	technique	Problèmes de compatibilité entre des codes open source	Prise de retard sur une tâche	Prise de retard sur les déliverables si la tâche est située sur un chemin critique	Moyenne	Moyen
12	technique	Mauvaise gestion du planning	Etre incapable de livrer à temps une maquette fonctionnelle	Echec du projet	Faible	Grave
13	technique	Impossibilité d'accès à des matériels ou sources d'infomation due au télé-travail	Etre incapable de livrer à temps une maquette fonctionnelle	Prise de retard sur les déliverables Echec du projet	Faible	Grave
14	technique/humain	Oubli de suavegarder Bug informatique Perte ou vol de matériel	Perte d'une partie du code	Prise de retard sur la réalisation du livrable	Moyenne	Grave

Fig. 1.5 – Liste des risques

Par la suite, nous les avons représentés graphiquement sur une matrice des risques standardisée. Celle-ci nous a permis de vérifier l'absence de risque critique pour la réalisation de notre projet (zone rouge dans la matrice exposée ci-dessous). Les risques identifiés en orange sont à surveiller particulièrement.

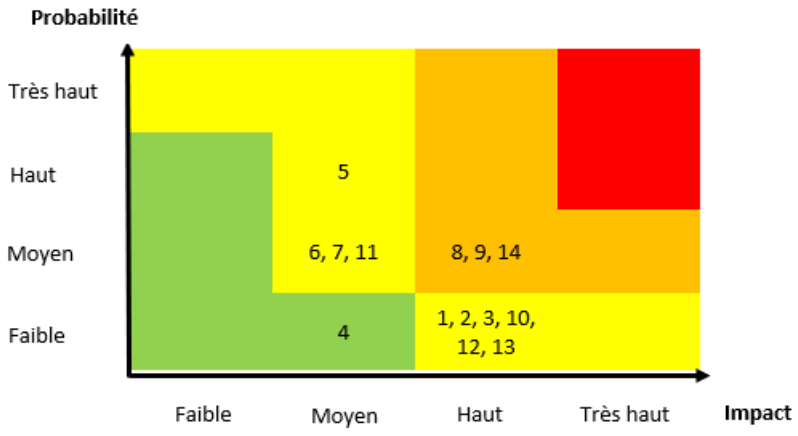

Fig. 1.6 – Matrice des risques

Pour conclure notre analyse, nous avons élaboré un ensemble de solutions préventives et curatives pour pallier à ces risques.

Num.	Actions préventives	Actions curatives	Responsable de l'action
1	Convenir du prochain créneau de réunion en face à face	Contacter une personne du même service que celui où le client travaille afin d'en comprendre les raisons et trouver	Chef de projet
2	Trace écrite récapitulative après chaque réunion	Discussion avec le client pour limiter les changements	Chef de projet
3	Réunions de médiation Evènements de teambuilding	Discuter en dehors du PIE afin de trouver une solution au problème	Chef de projet
4	Réunions hebdomadaires sur l'avancée du travail de chacun	Comprendre les raisons et répartir les heures de travail d'une nouvelle façon	Chef de projet
5	Avoir un besoin très clairement défini avant de commencer tout développement Faire un point d'avancement régulier avec le client	Se faire aider par le client	Chef de projet
6	Répartition claire des tâches à effectuer entre les membres de l'équipe en amont du projet Définition précise du planning	Redistribuer les tâches entre les membres de l'équipe Définir un nouveau planning	Chef de projet
7	Granulariser les différentes tâches du Gantt afin de faire un suivi détaillé du volume horaire de chaque tâche	Si possible, réallocation des heures passées sur chaque tâche Sinon, faire des heures supplémentaires	Chef de projet
8	Utilisation de Protégé afin de s'assurer de la compatibilité tout au long du projet	Changer le langage de programmation	Responsable adaptation du code
9	Suivre des formations et tutoriels	Demander de l'aide à une personne compétente pour ce type de langage	Responsable adaptation du code
10	Tests intermédiaires	Utiliser le versioning afin de retrouver une maquête qui fonctionne	Responsable adaptation du code
11	Utilisation de Protégé afin de s'assurer de la compatibilité tout au long du projet	Homogénéiser le langage de programmation	Responsable adaptation du code
12	Suivre les indicateurs du projet afin d'anticiper un éventuel retard	Utiliser le versioning afin de retrouver une maquête qui fonctionne	Chef de projet
13	Enumérer en amont du développement l'ensemble des matériels et sources d'information nécessaires à la réalisation du projet	Etablir une nouvelle stratégie pour atteindre l'objectif du projet	Chef de projet
14	Sauvegardes régulières Utilisation d'un logiciel de versioning comme GitHub	Allouer des heures à la partie code afin de réécrire le code perdu	Chef de projet

Fig. 1.7 – Liste des actions préventives et curatives

Le confinement mis en place à la Toussaint 2020 a amené l'équipe à travailler et à se coordonner entièrement à distance de novembre à fin janvier. Ce contexte, qui nous a obligé à progresser en mode dégradé, nous a fait revoir les risques identifiés en termes de probabilité d'occurence et d'impact. Ainsi, trois risques ont été mis à jour :

— Le risque 2, lié à la redéfinition tardive des objectifs : le télé-travail nous a fait émettre des doutes quant à la faisabilité de certains objectifs fixés au début du projet, du fait que l'accès à certains matériels de l'école indispensables à leur réalisation ainsi que l'obtention de conseils de la part d'enseignants-chercheurs nous étaient limités. Nous avons donc passé la probabilité d'occurence de ce risque de 'faible' à 'moyen' et son impact de 'haut' à 'très haut', le temps avançant agravant l'effet d'une potentielle redéfinition des objectifs.

— Le risque 12, lié à une mauvaise gestion du planning. La période confinée de novembre/décembre a eu tendance à impacter négativement la motivation de l'équipe. Lors de cette période, nous avons pris du retard sur le planning fixé initialement. Bien que nous ayons finalement rattrapé ce retard, cette expérience nous a poussé à passer la probabilité d'occurence du risque 12 de 'faible' à 'moyen', et son impact de 'haut' à 'très haut'.

— Le risque 13, lié à l'impossibilité d'accès à des matériels ou sources d'information à cause du travail à distance. Cette difficulté d'accès nous a ralenti dans l'implémentation de la commande vocale et du NLP sur la maquette. Nous avons donc passé la probabilité d'occurence de ce risque de 'faible' à 'moyen' et son impact de 'haut' à 'très haut'.

Cette mise à jour nous conduit à une nouvelle matrice des risques :

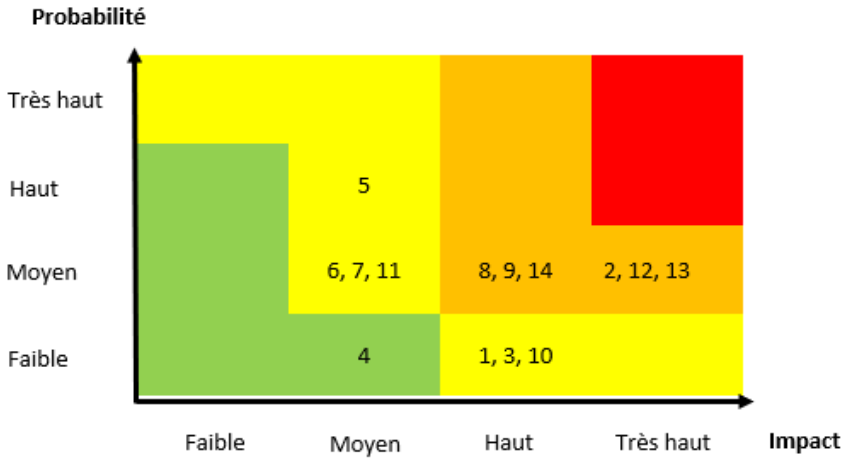

Fig. 1.8 – Matrice des risques mise à jour

Chapitre 2

Etat de l'art

Dans la première version du projet menée lors de l'année 2019/2020, l'état de l'art n'a pas été réalisé. C'est pourquoi nous allons y dédier une section ici. Nous détaillerons plusieurs types de technologies qui permettent une assistance intelligente à des tâches demandeuses (ou non) en attention.

2.1 Les assistants domestiques

Bien connu du grand public, les assistants domestiques sont une technologie qui pourrait potentiellement réaliser aisément une partie du travail demandé. En effet, la reconnaissance vocale, l'interprétation intelligente, la recherche et la restitution d'information sont très avancées sur ces produits. On peut mentionner par exemple le produit Amazon "ALEXA".
L'exemple choisi ici, "ALEXA", possède une palette de fonctions et de compétences permettant de faciliter le quotidien de ses utilisateurs. Cependant, tout le code consituant sa technologie s'exécute sur le cloud. Il n'est donc pas possible d'imaginer cette technologie en-ligne appliquée à notre problème.

2.2 Systèmes embarqués : automobile

Le secteur automobile fait sensation depuis plusieurs années, notament autour du phénomène TESLA qui prend de plus en plus d'ampleur. Ce cas est intéressant dans notre application comme nous sommes bien en présence d'une technologie d'assistance au pilotage (automobile) embarquée. Dans son ensemble, les voitures permettent une assistance à travers plusieurs fonctions, toutes pouvant être demandées (oralement pour la plupart) au système :

- commande des fonctionnalités classiques d'une voiture (vitres, climatisation, sièges,...) ;
- commande de type "confort" comme mettre de la musique, une radio, une vidéo (à l'arrêt) ;
- commande de pilotage automatique "autopilot" permettant de moins impliquer l'humain dans le pilotage.

La différence dans les fonctionnalités proposées est qu'ici, le code est exécuté à l'aide d'un système appelé "ASIC : Application Specific Integrated Circuit" et non sur un cloud. Cette technologie est principalement utilisée pour l'exécution du code en lien avec les fonctions d'autopilotage. Pour ce qui est du GPS, la radio, les musiques, le système est équipé d'une antenne 4G [2]. La technologie n'est donc pas totalement applicable mais a du potentiel pour les tâches concernées ici en avionique.

2.3 Airbus Single Pilot Operation

Avant la période Covid, la pénurie de pilote se faisait sentir. Le trafic aérien était en constante évolution et plus de 500 000 pilotes étaient nécessaires pour subvenir à ces besoin dans les 20 prochaines années au niveau mondial. Il est donc né un intérêt significatif de la part des avionneurs pour trouver une alternative économique. Dés lors, Airbus ne se cache pas en ce qui concerne ses recherches dans le domaine de l'automatisation des vols de ses avions [3]. L'ATTOL (Autonomous Taxi, Take Off and Landing) et le DISCO (Disruptive Cockpit) démontrent que des moyens financiers conséquents sont déployés pour individualiser le plus rapidement possible le cockpit.

ATTOL : Ce programme, comme son nom l'indique, vise à automatiser totalement certaines phases clé d'un vol : le roulage, le décollage et l'atterissage. En janvier 2020, Airbus a pu démontrer un décollage entièrement géré par l'auto-pilote. Démarré en Juin 2018, le programme s'est considérablement développé lors de la crise et les phases de vol sont à ce jour toutes opérationnelles. Le développement de cette technologie vise à diminuer la charge de travail du pilote et, d'une certaine façon, faire face à la pénurie de pilote à venir.

DISCO : Moins abouti pour le moment que le projet précédent, le programme DISCO vise à n'avoir plus qu'un seul pilote dans le cockpit. En plein développement, moins d'information sont disponibles quant à l'avancée des recherches. Les éléments bloquants ne sont pas les mêmes que pour l'automatisation d'un vol. Il fait pouvoir gérer le repos du pilote, monitorer son niveau d'engagement à chaque instant pour détecter une déconcentration, et être capable de l'assister dans les tâches accessibles. C'est un des rôles de notre assistant. Les prévisions pour des cockpits monopilotes estiment une date aux alentours de 2023 pour équiper les avions d'Airbus avec cette technologie, selon leur département ingéniérie [3].

2.4 Intérêt du projet

On l'a remarqué à travers les précédentes analyses : aucune technologie n'est aujourd'hui capable d'alléger la charge de travail du pilote tout en étant facilement implémentable au sein des systèmes existants. Toutefois, beaucoup de moyens sont développés pour avoir un système monopilote abouti et il n'y aurait pas grand intérêt à vouloir imiter les grands avionneurs. Il est alors légitime de se demander si notre projet d'assistance est encore pertinent dans le monde aéronautique aujourd'hui.

L'un des avantages de notre méthode de développement est de pouvoir aboutir à une assistance efficace implémentable sur les cockpits déjà en vol. En effet, le software en cours de développement ne nécessite aucune installation autre que logiciel. On peut cependant imaginer sur les futurs cockpits en développement l'intégration d'éléments comme des écrans dédiés uniquement à l'assistant ou un système de monitoring de l'attention du pilote.

Le ratio $\frac{assistance}{cout}$ de notre solution rendrait accessible le système à des compagnies aériennes souhaitant soulager leurs pilotes à court terme, par exemple à l'aide d'une tablette portable. Les sociétés de location de jets d'affaires sont également concernées : l'un de leurs arguments commerciaux reste la sécurité et le professionalisme des pilotes, alors renforcé par l'assistant virtuel. D'autre part, dans une version assez aboutie, le système pourrait permettre d'avoir un co-pilote moins qualifié et donc de réaliser certaines économies. Enfin, bien que notre projet se limite à l'aviation civile, gardons en tête l'activité militaire de Dassault Aviation. L'ontologie d'un avion de combat n'est pas la même que celle d'un jet d'affaire, mais la solution logicielle d'un assistant pilote pourrait être la même à bord du Rafale que dans le tout nouveau Falcon 6X.

Chapitre 3

Développement des solutions

Dans le respect des objectifs fixés lors du démarrage du projet avec le client, nous avons développé l'amélioration de l'assistant virtuel sur trois grands axes, le but étant d'avoir un assistant virtuel plus complet et plus simple à utiliser pour les pilotes.

1. Elargissement et étayage de l'ontologie ;
2. Intégration d'un moyen de formulation des requêtes par reconnaissance vocale ;
3. Mise en place d'une programmation en langage naturel (NLP).

3.1 Elargissement de l'ontologie

Dans le cadre de l'amélioration du projet entammé au cours de l'année 2019/2020, nous avons souhaité implémenter un ensemble plus riche de cas d'utilisation et de requêtes pouvant être formulées. Pour ce faire, nous avons réalisé des entretiens avec deux pilotes de lignes, co-pilotes sur A320 chez Air Francce et Easyjet. Ces entretiens nous ont permis d'identifier de nouveaux cas d'utilisation durant lesquels les pilotes sont preneurs d'assistance. Les sections ci-dessous répertorient ces nouveaux cas d'utilisation.

3.1.1 Scénarii implémentables

Déroutement : Le scénario ayant particulièrement été identifié comme situation nécessitant une assistance spécifique fut le déroutement. C'est une phase à forte charge mentale, souvent stressante et très demandeuse en termes de concentration, pendant laquelle les pilotes ont besoin d'obtenir des réponses rapides et précises sur leurs interrogations. Les idées évoquées lors des entretiens concernant l'assistance potentielle que pouvait fournir notre maquette lors d'un déroutement sont les suivantes :

1. Obtenir un rendu visuel sur les terrains les plus proches, avec si possible des informations sur la longueur de piste disponible et les capacités d'accueil propres à chaque terrain ;
2. Avoir une aide concernant le choix du terrain sur lequel se diriger, basée par exemple sur la direction du vent en permettant une approche avec le vent le plus possible de face, sur la longueur de piste dans le cas de volets ou reverses en panne qui induiraient un atterrissage long, ou encore sur la disponibilité des pompiers ou des urgences dans le cas d'un feu à bord ou d'un malaise.
3. Pour chaque terrain, avoir accès aux METAR, TAF, fréquence de l'ATIS, NOTAM, orientation des pistes ;

4. Obtenir, en fonction de la panne, notamment dans le cas de volets ou de reverses en panne, la distance d'atterrissage recalculée ;
5. Pour un terrain spécifique, pouvoir afficher la carte d'approche et les procédures propres à l'aérodrome ;
6. Pouvoir afficher les capacités de RFFS de chaque aérodrome (RFFS : Rescue and Fire Fighting Services) ;
7. En cas de pannes particulières, notamment volets ou trains d'atterrissage bloqués en position sortie, recalculer la consommation.

La présentation de ces informations au pilote est un sujet crucial pour que l'assitant virtuel soit ergonomique. Si certaines informations peuvent être lues oralement au pilote (météo, distance d'atterrissage, etc.), d'autres ne epuvent qu'êtres affichées. C'est le cas par exemple des cartes. De plus, les pilotes sont demandeurs d'éléments de décision mais pas d'un choix pris par l'intelligence artificielle. Ils souhaitent avoir à disposition tous les éléments pour choisir le meilleur déroutement, et ceci nécessite un affichage clair qui risque d'être surchargé si l'IHM n'est pas pensée en conséquence.

Les Checklists : Le souhait de pouvoir effectuer l'ensemble des checklists via notre IHM a également été formulé par les deux pilotes. Idéalement, l'ensemble des checklists propres à l'appareil devraient être disponibles numériquement dans la base de données. Jusqu'à maintenant, sur Airbus 320, l'ensemble des checklists est sous format papier. Une idée intéressante qui permettrait de mettre la reconnaissance vocale des requêtes à profit serait que lorqu'une checklist est effectuée par l'équipage, au moment de l'énumération de chaque item par les pilotes, l'IHM soit en mesure de détecter quel item est en train d'être vérifié et s'il a été réalisé correctement. Plus concrètement, un pilote pourrait appeler une checklist spécifique via l'IHM, l'interface afficherait la checklist demandée, en faisant apparaître les items un à un et en ne passant à l'item suivant que lorsqu'elle a détecté que l'item en cours de vérification a été checké de la bonne manière vocalement. Aussi, les procédures en cas de pannes sont sous format papier. Un accès rapide à l'information est préférable lors des phases critiques et leur implémentation numérique serait un avantage.

Météo sous forme de cartes : Actuellement, la maquette ne propose que des METAR ou des TAF. Une proposition des pilotes interrogés serait d'avoir numériquement à disposition des cartes TEMSI ainsi que des cartes WINTEM à jour afin d'améliorer l'approche des terrains.

Procédures complexes : Certaines procédures recquièrent une recherche laborieuse dans la documentation de l'avion de la part des pilotes. Par exemple, les procédures de dégivrage nécessitent de prendre en compte un nombre important de paramètres pour décider quel produit dégivrant demander aux équipes de l'aéroport. Les pilotes aimeraient donc pouvoir demander à l'assistant de déterminer seul le produit à utiliser en fonction de la température, l'humidité, etc. Ce genre de tâche simple à automatiser et pourtant demandeuse en ressources attentionnelles serait une vraie valeur ajoutée de notre assistant.

MEL (Minimum Equipment List) : La MEL répertorie l'équipement fonctionnel minimal que l'avion doit présenter pour être légalement autorisé à effectuer un vol commercial. Il arrive que la panne de certains équipements pose un réel questionnement au moment du départ sur l'autorisation légale d'effectuer un vol. La MEL constitue un document lourd, difficilement lisible et dans lequel il est compliqué de se retrouver. Les pilotes demandent donc l'implémentation numérique de la MEL dans l'IHM, pour avoir une possibilité de recherche simple et rapide.

Déroutements en aviation d'affaire : Cette situation est propre à l'aviation d'affaire. Dans certains cas, un déroutement peut être spontanément demandé par les passagers. Le pilote doit alors décider si la nouvelle destination demandée est accessible, que ce soit en termes de capacités (carburant restant, longueur de piste disponible à destination, météo...), qu'en termes de logistique (horaires d'ouvertures de l'aéroport, équipements de radionavigation disponibles sur le terrain de destination, notamment pour les atterrissages de nuit et les approches ILS, le carburant disponible à destination pour le ravitaillement, les capacités de handling de l'aéroport...). Les pilotes aimeraient donc avoir l'ensemble de ces informations accessibles via l'IHM.

CPDLC (Controller Pilot Data-Link Communication) : Le CPDLC permet aux pilotes de communiquer avec le contrôleur par message écrit, afin de ne pas encombrer la fréquence. Cette fonctionnalité pourrait être disponible sur l'IHM. Un processus de reconnaissance vocale représente un avantage pour cette fonctionnalité car il permettrait aux pilotes d'énoncer leur demande oralement et non par écrit, ce qui représente un gain de temps et permet de libérer les mains pour les actions de pilotage.

3.1.2 Améliorations visées

La structure du programme nous a été imposée par le projet de l'année précédente. Il y a donc des tâches qui seront plus facilement implémentables que d'autres. Nous avons choisi de concentrer nos efforts sur les points suivants :

1. Implémentation de la fonction "Checklist Review" : Le pilote appelle la checklist et celle-ci se développe une itération après l'autre après chaque retour vocal du pilote.
2. Déroutements propres à l'aviation d'affaire : Implémentation des capacités de handling des aéroports, du carburant disponible sur site, les capacités RFFS disponibles, la longueur et l'orientation des pistes.
3. CPDLC : Le pilote appelle la fonction pour ensuite envoyer un message au contrôleur de son choix qui sera transmis tel quel par écrit.

La raison du choix de ces améliorations est leur faisabilité au regard du code existant. Le contexte choisi ne permet pas d'avoir une implémentation optimale et réaliste de la fonction "déroutement", par exemple. Cela demanderait le développement d'une base de donnée conséquente et nous n'avons pas les ressources pour le faire. A notre échelle, nous aurions pu réaliser une démonstration de la fonction sur un déroutement prévu au cours d'un vol planifié, mais ce ne serait pas très pertinent.
La "Checklist Review" est une capacité de l'IHM qui a suscité l'intérêt des deux pilotes. C'est une fonctionnalité réalisable qui pourrait apporter une réelle plus value à la maquette. Les checklist n'étaient pas présentes jusqu'ici et la possibilité de coupler cette fonctionnalité à la reconnaissance vocale afin d'assurer que chaque item d'une checklist ait bien été vérifié constitue un vrai plus en terme de sécurité. Enfin, l'implémentation du déroutement propre à l'aviation d'affaire présente une bonne faisabilité puisqu'elle consiste en une simple mise à jour de l'ontologie déjà existante.

3.1.3 Implémentation

Dans cette section, nous présentons l'implémentation des fonctionnalités citées précédemment afin de rendre la maquette plus ergonomique et plus adaptée aux situations rencontrées dans un cockpit. Nous y retrouverons les détails d'implémentation des fonctionnalités supplémentaires que nous avons développées, couplées avec les axes d'amélioration de départ.

Checklist Review

Nous avons implémenté cette fonction en suivant à la lettre les souhaits des pilotes interrogés. Les étapes sont les suivantes :

1. Le pilote appelle la Checklist (Ex : "Give me the Before Take-off Checklist") ;
2. La maquette renvoie le premier item de la checklist avec le mot que doit prononcer le pilote pour confirmer qu'il à vérifier l'item (Ex : Flaps, say 'full') ;
3. Le pilote prononce le mot de vérification approprié ;
4. Itération du procédé présenté pour chaque item de la checklist ;
5. Une fois l'ensemble des items de la checklist passé en revue, la maquette affiche "Before Take-Off checklist Completed".

On peut voir en figure 3.1 un exemple d'affichage de l'IHM lors de l'utilisation de cette fonctionnalité.

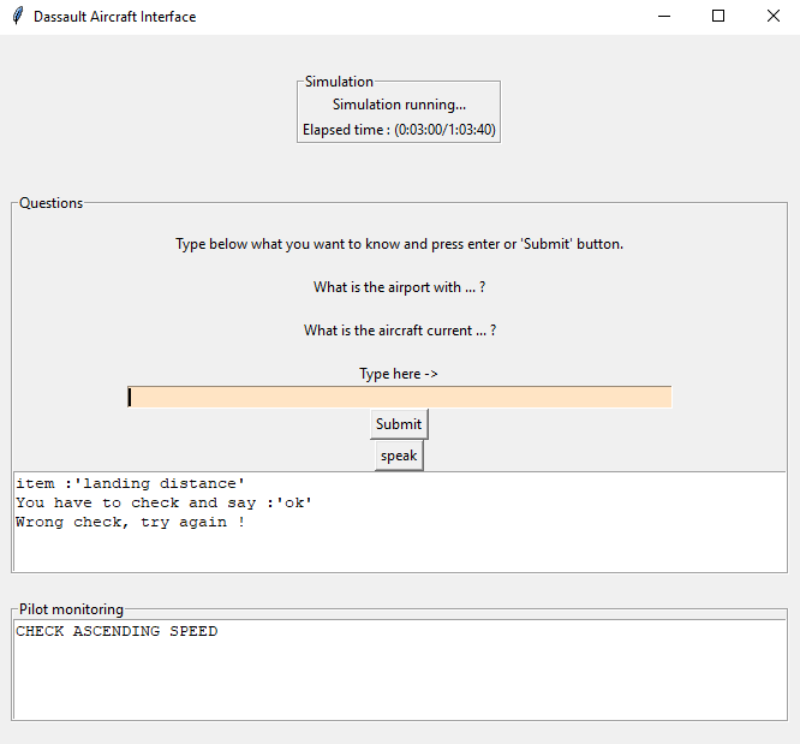

Fig. 3.1 – Checklist Review

Déroutement propre à l'aviation d'affaire

On rappelle que cette fonctionnalité consiste à rendre disponible des informations dont les pilotes d'aviation d'affaire ont besoin pour décider de la faisabilité d'un vol, le plus souvent vers un petit terrain ne présentant pas les mêmes équipements et capacités d'accueil que les grands aéroports. Nous avons implémenté, pour chaque aéroport, les informations suivantes :

1. Les horaires d'ouverture de l'aéroport ;
2. Les horaires d'ouverture du contrôle aérien ;

3. Le carburant disponible au niveau de l'aéroport ;
4. La largeur des taxiways de l'aéroport ;
5. Les capacités de handling de l'aéroport.

Chacune de ces informations peut être demandée oralement au cours d'un vol. Par exemple, une demande de l'essence disponible à Paris pzut être formulée : "what is the type of fuel available at Paris-Orly ?"

CPDLC

On rappelle que le CPDLC consiste en un système de d'échange pilote-contrôleur par essage écrit. L'utilisation de cette fonctionnalité se fait par étapes :

1. Le pilote demande vocalement à parler à un controleur spécifique, par exemple : 'Can I speak to the air controller of Toulouse ?' ;
2. L'IHM reconnaît la demande et va chercher la fréquence correspondante, il affiche alors par exemple : "You are speaking to Toulouse ATC on frequency 181.1, say your message" ;
3. Le pilote énonce vocalement le message qu'il veut transmettre ;
4. L'IHM envoie le message retranscrit et affiche : "message sent" ;

On peut voir en figure 3.2 une étape du procédé.

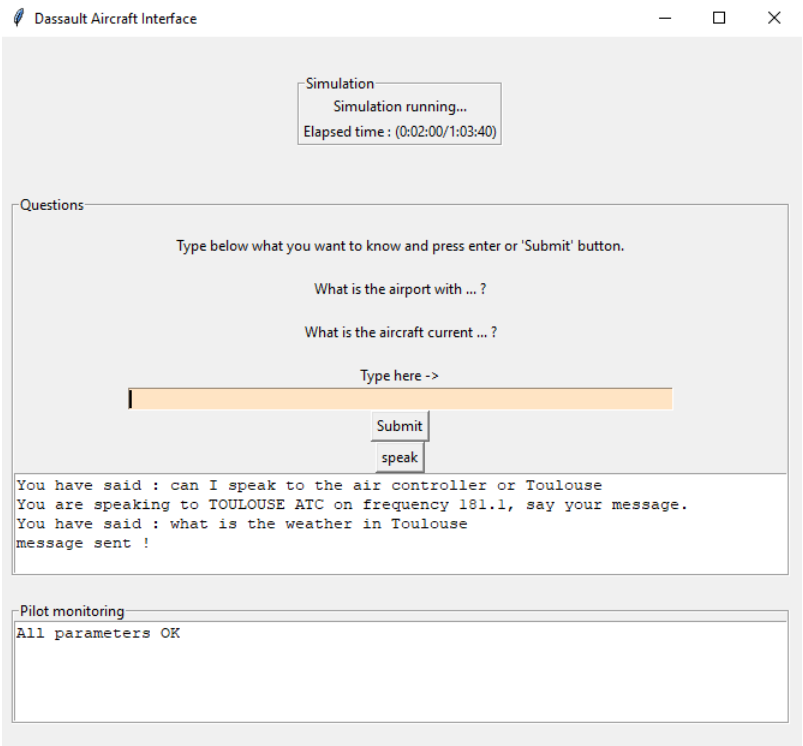

Fig. 3.2 – CPDLC

3.2 Programation en langage naturel (NLP)

3.2.1 Reconnaissance du langage naturel

La maquette obtenue à l'issue du PIE 2020 imposait une formulation stricte des requêtes, celles-ci étant traitées par une succession de if et else vérifiant la présence ou non de mots-clés dans les phrases. Outre la lourdeur du code, cette implémentation était très limitée et ne pouvait être utilisée dans un environnement réel de travail. L'objectif principal de notre travail était de donner aux pilotes la possibilité de poser n'importe quelle question à leur manière, en anglais. Nous sommes ici face à un problème de NLP (Natural Langage Processing), et plus particulièrement de NLU (Natural Langage Understanding). Ce champs de recherche est particulièrement plebiscité actuellement pour optimiser les chatbots, de plus en plus présents pour offrir différents services [4]. Les techniques existantes sont particulièrement variées, mais reposent généralement sur la vectorisation des mots, appuyée sur une base de données de l'ensemble des mots de la langue concernée.

3.2.2 Solutions disponibles

Il existe un très grand nombre de bibliothèques de Natural Langage Processing disponibles en Open-Source, et une grande partie est implémentée en Python. La première état a donc été de choisir la plus adaptée à notre besoin et faisant une étude de l'état de l'art disponible. Tokenizers est la plus récente des bibliothèques disponibles aujourd'hui. Elle est particulièrement performante dans la tokenization des mots, et reconnue pour sa rapidite (20 secondes pour traiter un GB de texte). SpaCy est la solution la plus répandue en Python par sa faculté à s'intéger au sein d'autres bibliothèques de machine learning. Dans un premier temps elle nous a semblé intéressante pour notre utilisation. Le framework BERT de Google est une solution particulièrement puissante par son utilisation en ligne et un entraînement reposant sur une base de données gigantesque. Mais c'est vers NLTK (Natural Langage Toolkit) que nous nous sommes tournés pour les premiers tests. C'est la bibliothèque la plus répandue et une grande quantité d'aide est disponible en plus de la documentation déjà très complète.

Dés le début de l'implémentation, nous avons cependant réalisé que l'outil était bien plus lourd que nécessaire, et qu'il serait compliqué d'entraîner notre propre modèle. D'abord, la bibliothèque couvre tout le NLP, quand nous n'avions besoin que de la partie Understanding (NLU) et non pas Generating (NLG). Les modèles déjà entraînés n'étaient pas adaptés à notre utilisation, et nous ne disposions pas d'un dataste assez grand pour entraîner notre momdèle. En effet, le cahier des charges nous plaçait dans un entre-deux : le nombre de requêtes que peut poser un pilote en vol reste limité à sa situation et celle de son appareil. Chaque personne s'exprime différemment, mais il n'y a pas une infinité de formulations pour chaque question, particulièrement dans un domaine avec un vocabulaire si typé que l'aéronautique. Nous nous sommes donc résolus à ne pas utiliser de technique de machine learning ou d'apprentissage quelconque, mais plutôt réaliser un set de dictionnaires qui auraient permis de gérer les requêtes prédéfinis avec beaucoup de simplicité et d'efficacité, mais moins de fiabilité.

3.2.3 Snips NLU

Cependant, quelques recherches supplémentaires nous ont permis de trouver la bibliothèque SnipsNLU, correspondant parfaitement à notre besoin. Celle-ci permet la compréhension du langage naturel en langue anglaise, avec une possibilité très simple d'entraînement et d'orientation sur le domaine souhaité

[5]. Comme on peut le voir sur la figure 3.3, cette solution a une précision qui ravalise avec les références du domaine, les outils de Microsoft (Luis.ai) ou de Google (Api.ai), mais elle se démarque par sa légereté et sa facilité d'adaptation à un domaine linguistique.

Fig. 3.3 – F1-score des différentes bibliothèques de référence en NLU

Pour entraîner le modèle sur le domaine d'intérêt, le concepteur complète un dataset avec les requêtes attendues et les mots-clés que l'utilisateur pourra prononcer. Le format yaml permet de compléter ce dataset manuellement et de manière intuitive, cela permet de l'enrichir au fur-et-à-mesure de l'utilisation. La syntaxe permettant de définir un type de requête à détecter est présenté en figure 5.2 et 5.1. La bibliothèque convertit ensuite ce fichier en JSON, sur lequel est entraîné le modèle.

3.2.4 Implémentation et résultats

Dés les premiers tests les résultats ont été très satisfaisants. Chaque question était classifiée selon un type de requête (demande de météo, d'aéroport, de checklist, etc.) et renvoyait les mots-clés nécessaires à l'élaboration de la réponse (élément météo demandé, aéroport concerné, etc.). En enrichissant le dataset pour couvrir la totalité des questions que le pilote peut poser, certains conflits sont apparus. Par exemple, lors de la demande du plus proche aéroport sur lequel le pilote peut atterrir, snipsNLU renvoyait parfois la checklist d'atterrissage, selon la manière dont la question était formulée.

La meilleure manière pour résoudre ce conflit est de mieux différencier les requêtes afin de réduire la probabilité de collision. Ceci peut-être fait par l'utilisation d'une phraséologie plus précise par exemple. Cela nous a semblé être la meilleure solution, cette technique étant déjà utilisée très couramment dans les communications aéronautiques. L'objectif reste de donner la plus grande liberté aux pilotes, mais les échanges avec des professionnels nous ont confirmé que ceux-ci n'étaient pas réticents à utiliser une phraséologie légèrement encadrée.

Ainsi, nous avons quitté une implémentation limitée de la compréhension du langage (enchaînement de if et else) pour s'adapter au langage naturel. La solution est particulièrement satisfaisante, mais nécessite une précision dans la construction du dataset ainsi qu'un certain cadre dans la manière de poser les questions. Malgré la qualité de l'entraînement du modèle, lorsque certains mots sont absents de la question il est sûr que l'intelligence artificielle ne l'interprêtera pas de la bonne manière.

En 2020, un projet de master a également porté sur le développement d'un assistant pilote virtuel [6]. L'équipe d'étudiants a particulièrement approfondi cet aspect de compréhension du langage naturel notamment en utilisant des méthodes POMDP pour améliorer la compréhension de la question suivante en fonction du dialogue actuel et de la situation globale. Ce projet a abouti à la création d'une bibliothèque que nous n'avons pas pu récupérer, mais qui, avec quelques adaptations, pourrait se révéler être très adaptée à notre besoin pour une implémentation future.

3.3 Reconnaissance vocale

Après entretien avec les pilotes, il est ressorti que la fonctionnalité de reconnaissance vocale apporterait une facilité d'utilisation indispensable à ce type d'assistance. Nous avions alors plusieurs possibilités pour l'implémenter. Le code s'exécutant en python, nous avons cherché des solutions qui seraient compatibles. Les possibilités sont les suivantes.

Librairie Port Audio, Py Audio et SpeechRecognition : La librairie Port Audio sur python est une librairie qui permet de traiter des entrées et sorties audio en temps réel. Couplée avec l'interface PyAudio qui fait le lien avec les entrées sur l'ordinateur et le code, l'application offre un traitement de la parole satisfaisant en temps de réponse. Enfin, la librairie SpeechRecognition s'occupe du traitement du signal vocal et de sa traduction en une chaîne de caractères. La librairie a été configurée en anglais pour coller avec la maquette. En moyenne, pour des phrases types de la communication aéronautique, le système reconnait la phrase en 2.1 secondes après la fin de l'élocution.

Autres librairies : Afin d'avoir un regard critique sur cette partie, nous avons comparé la première solution à une autre librairie disponible en OpenSource en python. Il existe plusieurs librairies liées à la détection vocale comme par exemple "inaSpeechSegmenter" ou encore "WebRTC VAD" qui peuvent être couplées à des méthodes pour obtenir un résultat convainquant mais cela n'est pas adapté à notre problème. L'intérêt de ce type de librairie est d'avoir un meilleur contrôle sur les paramètres permettant d'adapter la reconnaissance sonore en fonction de la nature du son.

Logiciels externes : Une autre solution serait de passer par un logiciel externe qui serait appelé par le programme. Il en existe plusieurs et pour cet exemple nous avons choisi Braina Pro. L'intérêt de passer par ces logiciels est l'aboutissement de la reconnaissance vocale qui peut être adaptée à plusieurs accents, voix, un grand répertoire de vocabulaire technique, juridique, etc.

Le choix pour la reconnaissance vocale s'est donc logiquement tourné vers l'ensemble PortAudio/SpeechRecognition. Les autres librairies de python étant trop complexes à mettre en place. Comparé à PortAudio, l'interfaçage de Braina Pro avec la maquette est trop lourd à implémenter. Après divers tests concluants, l'ensemble de ces librairies permet aussi bien de reconnaître un accent natif anglais qu'un accent français très prononcé [1]. Pour ce qui est du vocabulaire aéronautique, il n'y a pas non plus de

1. Tests réalisés avec un interprète professionnel.

problème particulier. D'autre part, une routine SpeechRecognition dans le programme python est très pragmatique. Une fois implémentée, la routine retourne relativement facilement un élément String qui peut par la suite être traité sans perturber la structure du code.

3.3.1 Robustesse de la solution

Bien que la librairie soit très performante en plus d'être gratuite, la reconnaissance de l'énonciation des lettres a posé quelques soucis. Il y a plusieurs situations où le pilote doit énoncer une série de lettres comme lorsqu'il énonce le code OACI d'un aérodrome. Si on prend l'exemple avec le code OACI de Toulouse-Blagnac, lorsque l'on énonce "LFBO" oralement, la librairie peut parfois comprendre "LFPO" ou même rater une lettre.
Pour y remédier, nous avons choisi d'appeler les aérodromes par leur nom complet. Par exemple, LFPO sera appelé "Paris-ORLY", les noms de ville étant associés ensuite aux aéroports par la maquette. Certains autres éléments peuent être difficilement reconnus : de manière générale, les mots isolés comme "check" dans l'utilisation des checklists peut poser problème. Pour améliorer la précision de l'assistant, on compte d'une part sur la compréhension du langage naturel pour améliorer la compréhension selon le contexte, et d'autre part sur le pilote pour choisir les bons mots et les énoncer clairement.

3.4 Interactions pilote - assistant

Nous avions mentionné au début du projet qu'il pouvait être intéressant d'effectuer des expériences avec la maquette aboutie afin d'optimiser les échanges homme-machine. En accord avec le client cet objectif n'avait pas été défini comme prioritaire. Le temps nous a effectivement manqué pour réaliser une campagne de test avec un échantillon représentatif de pilotes sur un simulateur présentant un environnement écologique. Cependant dés lors que la maquette était fonctionnelle nous avons utilisé une méthode itérative avec des tests simples pour nous assurer de l'ergonomie de la solution.

Au cours de son développement, nous avons jugé utile d'ajouter une réponse vocale à l'assistant dans certaines situation. Cette idée nous a été amenée lors des entretiens pilotes et s'est avérée réalisable. Intuitivement, on peut supposer que certaines informations sont plus agréables à recevoir oralement plutôt que par écrit. Cependant, cela ne veut pas dire que la solution est pertinente en toute situation. En effet, l'effort mis dans l'écoute active diffère fortement et les ressources cognitives varient en fonction des personnes [7]. Dans cette logique, nous avons implémenté une réponse vocale dans les situations suivantes :

- Au cours des Check-lists ;
- Lors d'une demande de déroutement.

Cette fonctionnalité a été intégrée dans le but de mettre en forme la technologie de la réponse vocale sur la maquette et de proposer une piste d'amélioration au projet qui sera alors à étudier plus en profondeur. Nous ne proposons pas ici une alternative aux informations reçues directement par écrit, nous posons uniquement une problématique qui mérite une étude complète et approfondie.

3.4.1 Solution pour le développement de la réponse vocale

La réponse vocale de l'assistant repose sur le concept du text-to-speech (TTS) qui est le processus de conversion de mots en une forme audio sous forme vocale. Premièrement, l'outil prend un texte d'entrée, en sortie de la maquette dans notre cas. Ensuite, à l'aide de méthodes de traitement du langage naturel, l'algorithme comprend la linguistique utilisée et effectue une inférence logique sur le texte. Ce texte traité est passé dans un bloc suivant où le traitement du signal numérique est effectué. Le texte traité est finalement converti en un prononciation orale. Pour mieux visualiser les étapes, le schéma de principe est représenté en figure 3.4.

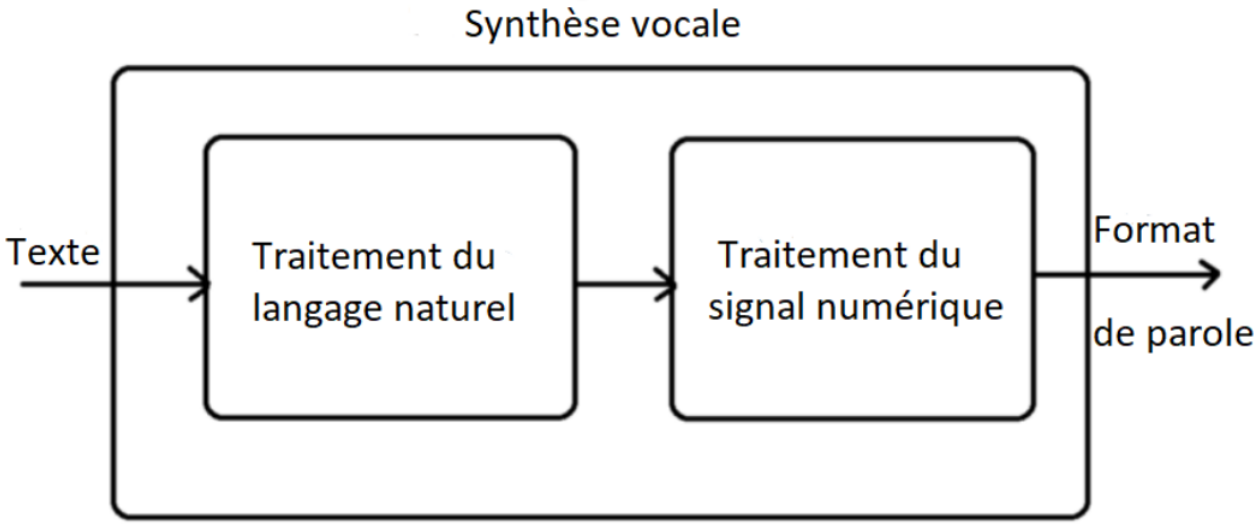

Fig. 3.4 – Etapes pour la synthèse vocale

Grâce à Python et au module gTTS (Google Text-to-Speech), ce processus est simplifié en quelques lignes de code. À partir du diagramme, nous pouvons comprendre que le texte transmis est d'abord prétraité à l'aide du traitement du langage naturel, puis converti en parole en utilisant le traitement du signal numérique.

La bibliothèque python gTTS est un outil pour échanger avec l'interface de programmation de synthèse vocale de Google Translate. Nous importons donc la bibliothèque gTTS du module qui peut être utilisée pour la traduction vocale. Le module gTTS peut également être utilisé dans d'autres langues telles que le français, l'allemand, l'hindi, etc.

3.5 Architecture finale de la solution

Une fois toutes les fonctionnalités précédentes implémentées, on obtient l'architecture logicielle définie sur l'image 3.5. Les données sont réparties dans différents dossiers locaux, et chaque set de données est exploité par un script en particulier, lui-même appelé par main.py. On remarque que la reconnaissance vocale comme la réponse orale utilisent les données présentes sur les serveurs de Google et nécessitent donc une connexion à Internet. Cela est acceptable pour la maquette mais il faut avoir en tête cette limitation avant de penser à une implémentation en environnement écologique.

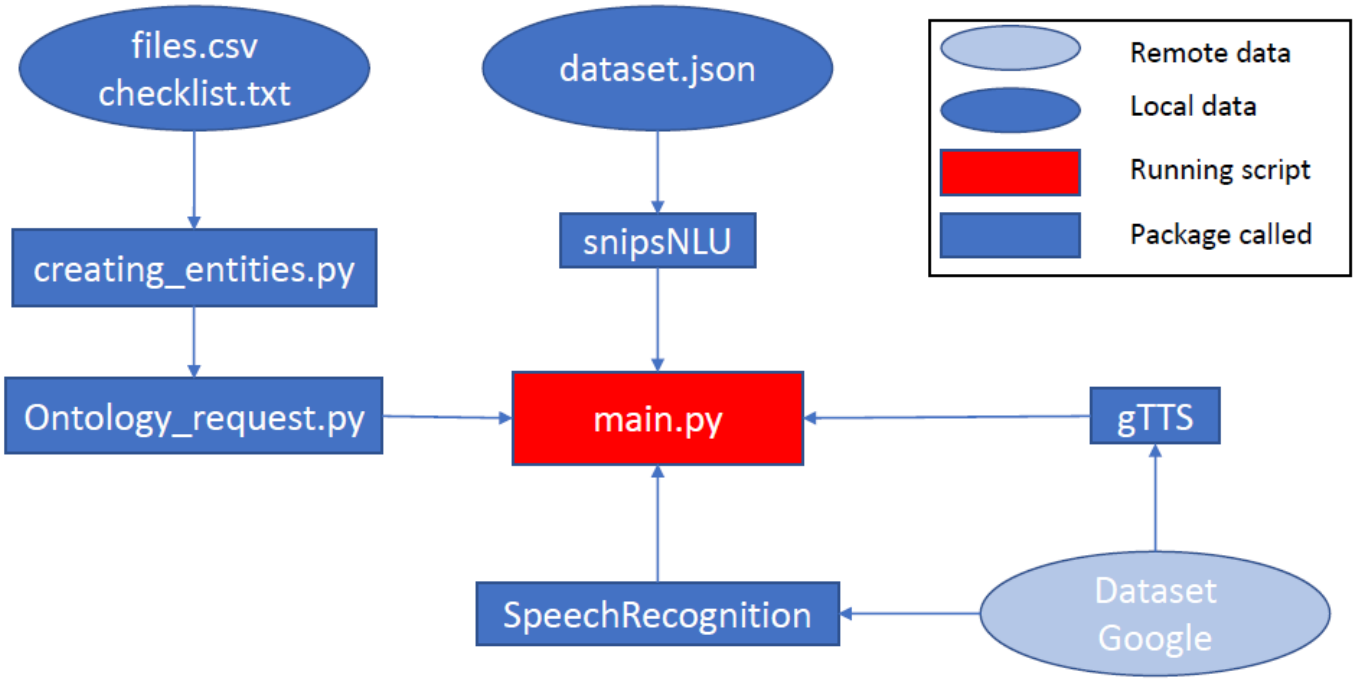

Fig. 3.5 – Architecture finale du logiciel

Chapitre 4

Utilisation de la maquette

Après la présentation des fonctionnalités ajoutées à la maquette, nous souhaitons dresser un premier bilan de son utilisation. Une fois la maquette fonctionnelle et satisfaisante, nous avons recontacté les pilotes interviwés en début d'année pour avoir leur retour profressionnel. Nous avons ensuite réalisé une étude de performance de la maquette puis nous avons détaillé quelques pistes d'amélioration.

4.1 Feedback pilote

Dans le but de reboucler notre démarche, nous avons fait tester notre maquette améliorée par les pilotes qui nous avaient conseillés pour étayer les scénarios d'utilisation de la maquette. L'objectif ici est d'avoir un retour d'expérience par des potentiels utilisateurs finaux. Lors des interviews, les pilotes se sont fait une idée de ce à quoi ressemblerait cette maquette, nous attendions donc des remarques sur la forme. Ces pilotes nous ont également détaillé des fonctionnalités qu'ils aimeraient retrouver, c'est alors l'occasion de savoir si ces fonctionnalités répondent à leurs attentes. Les remarques ont été les suivantes :

— Le processus de requête et de réponse vocal implémenté sur la maquette a recueilli un très bon feedback. La possibilité de formuler les requettes à l'oral et non à l'écrit est un réel gain de temps et de concentration pour les pilotes. De plus, la possibilité de recevoir la plupart des réponses de la part de la maquette sous format audio permet aux pilotes de concentrer leur attention visuelle sur les instruments de vol et l'extérieur, apportant un avantage significatif en terme de sécurité.

— L'amélioration du traitement des requêtes, en particulier via l'implémentation du NLP qui autorise les pilotes à formuler leur demande sans être obligés de suivre une construction phraséologique précise a également été considérée comme un avantage significatif. Dans le même esprit que le système de réponse vocale, cette implémentation augmente la liberté des pilotes et leur permet de se concentrer en priorité sur le pilotage de l'avion, sans avoir une phraséologie précise à respecter.

— Le processus de checklist par reconnaissance vocal a également été bien accueilli. Cette amélioration présente un intérêt tout particulier pour une potentielle future génération d'avions mono-pilotes. En effet, jusqu'à maintenant, le travail en équipage permet lors de la réalisation d'une checklist qu'un pilote énonce chaque item tandis que l'autre vérifie l'item. Dans le cas d'un avion mono-pilote, l'IHM peut faire l'office de second pilote lors de la réalisation d'une checklist, puisqu'elle s'assure que chaque item a bien été énoncé et checké. La vérification automatique peut également être plus performante qu'un humain dans certaines conditions de charge de travail élevée et ainsi apporter en sécurité.

— L'implémentation du CFDLC sur la maquette est considérée comme un réel atout. De plus en plus d'aéroports s'équipent de ce système d'échange controleur-pilote par message écrit, l'intérêt étant de limiter l'encombrement des fréquences. Le désavantage du CFDLC réside dans la formulation des messages par écrit, qui présente une perte de temps significative pour les pilotes et une impossibilité de faire autre chose en même temps, notamment durant les phases de décollage et d'approche. L'IHM, en permettant la retranscription d'une formulation orale à l'écrit, permet d'exploiter au mieux ce système d'échanges.

— Un des bémols évoqués par les pilotes pointe le manque de robustesse du processus de formulation des requêtes par commande vocale. En effet, l'utilisation de termes propres à l'aéronautique, l'identification des aéroport par leur code OACI, l'énonciation d'une fréquence pour échanger avec un contrôleur ou encore la formulation de certaines requêtes par un mot unique sont des facteurs de mauvaise reconnaissance par le système de reconnaissance vocale. Ce manque de robustesse proscrit l'implémentation de ce système en conditions de vol réel aujourd'hui, les quelques incompréhensions faisant perdre plus de temps au pilote qu'il ne peut en gagner. Cependant, l'idée du processus a fait l'objet d'un intérêt tout particulier de la part des pilotes : l'enjeu sera de l'améliorer.

— Le second bémol évoqué concerne le manque de fonctionnalités exigeant une réponse visuelle de la part de l'IHM. Par exemple, en sitation de déroutement, les pilote souhaiteraient avoir un rendu visuel sur les terrains disponibles à proximité de leur position, pouvoir récupérer facilement la longueur de piste disponible, les capacités d'accueil propres au terrain en question, la fréquence radio pour contacter le contrôleur local, voire avoir visuellement un conseil concernant le terrain sur lequel il est préférable de se dérouter.

4.2 Performances de la maquette

À la demande du client, nous allons détailler ici les performances de la maquette au regard du temps d'exécution et de la mémoire RAM. L'étude vise à donner un ordre de grandeur de la consommation de ressource, sans avoir besoin d'une précision excessive.

4.2.1 Temps d'exécution

L'élément le plus gourmand en temps de traitement est la reconnaissance vocale. Sans elle, la maquette retourne les requêtes de façon pratiquement instantanée. Une des raisons pour lesquelles ce temps de traitement prend un certain temps est la prise de conscience de fin de phrase. C'est à dire que, en fin de phrase, le programme prend quelques secondes pour comprendre que la personne a fini de parler.
Ci-après, on retrouvera le tableau 4.1 qui reprend les différents temps de traitement, calculés à l'aide d'un timer implémenté directement dans le code. Les tests sont réalisés sur un ordinateur portable classique [1]. Ce temps est compté à partir du moment où la personne a fini de parler jusqu'à avoir une réponse écrite. Nous allons calculer ces performances en deux temps. Dans un premier temps, sans retour vocal de la part de la maquette et ensuite avec. Lorsque qu'il y aura un retour vocal, nous arrêterons le timer lorsque la réponse aura fini d'être énoncée. Le temps est inévitablement bien plus long à cause du format de la réponse mais ces données pourraient être utiles dans une démarche future. En effet, certaines informations pourraient être trop longues à recevoir oralement.

1. Dans notre cas, l'ordinateur était un ASUS Notebook équipé d'un processeur i5.

Type de requête	**Temps de traitement** [s]
Sans retour vocal	
Demande d'altitude	$8.02\ 10^{-1}$
CPDLC	1.3
Carburant restant	$8.44\ 10^{-1}$
Avec retour vocal	
Demande d'altitude	5.61
CPDLC	3.52
Carburant restant	4.38

TABLE 4.1 – Temps de traitement des différentes requêtes

Nous avons mentionné précédemment que la réponse vocale introduit une latence simplement par le fait que la maquette doit comprendre que la phrase est finie. Ce temps ne peut pas être pris en compte avec le timer implémenté. Pour quantifier cette latence, nous avons décidé de réaliser de nombreux tests à la main. La fiabilité n'est pas optimale mais sur un grand nombre de tests, on peut avoir une moyenne qui nous permettra de quantifier cette latence. Pour ces séries de tests, nous avons regroupé nos résultats sous plusieurs catégories que nous avons jugées pertinentes grâce à l'expérience que nous avons acquis en manipulant notre maquette. Ces catégories sont les mots, les phrases courtes et les phrases longues. Le tableau 4.2 ci-après reprend la différence entre le temps donné par le timer et le temps que nous avons calculé. Nous avons procédé à une trentaine de tests par catégorie.

Type de catégorie	$\Delta_{traitement} = t_{main} - t_{timer}$ [s]
Mot	2.23
Phrase courte	1.86
Phrase longue	1.92

TABLE 4.2 – Temps de traitement des différentes requêtes sans le timer ni réponse vocale

Nous voyons sur ce tableau que la latence semble être en grande partie introduite par le phénomène expliqué au paragraphe précédent. Celui-ci introduit environ 70 à 80% de la latence dans chacun des cas. La façon de mesurer n'est pas optimale mais suffisante pour en venir à cette conclusion. Ainsi, l'utilisation d'un bouton push-to-talk dans le cockpit permettrait un gain de temps conséquent par rapport à un simple click-to-talk. Cette solution semble être la bonne, un tel bouton étant déjà présent sur le yoke ou le minimanche des pilotes pour émettre à la radio. Dans un futur moins proche, on peut imaginer que l'assistant sera à l'écoute en permanence comme les assistants domestiques qui se déclenchent à l'écoute d'une phrase telle "Dis Siri" (Apple) ou "Ok Google" (Google Home). Cela implique d'avoir un bon traitement de l'environnement sonore du cockpit pour ne pas louper un appel.

4.2.2 Mémoire RAM utilisée

De la même façon que pour le temps d'exécution, nous allons ici détailler les ressources allouées pour différents types de requêtes. Le tableau 4.3 reprend les différents cas de figure. Les tests sont toujours effectués sur le même ordinateur que précédemment.

D'après nos estimations, la puissance de calcul nécessaire est accessible à tout type d'ordinateur. Cela

Type de requête	**Ressource RAM** [Mo]
Demande d'altitude	512
Checklist Review	513
Carburant restant	512

Table 4.3 – Ressources allouées pour les différentes requêtes

rend notre maquette très mobile. En effet, la mémoire RAM utilisée pour faire tourner la maquette est constante et varie autour de 512 Mo. La solution complète occupe un espace de l'ordre de la dizaine de mégas sur le disque, selon le volume de la base de donnée. Cette approximation permet d'imaginer notre assistant embarqué sur une tablette dans les cockpits actuel, avant d'être intégré aux calculateurs de vol dans les futurs avions.

4.3 Pistes d'amélioration

Le projet que nous avons mené était déjà dans la conitnuité d'un projet précédent et nous pensons que celui-ci pourrait encore mériter un développement orienté sur certains points. Le but de cette section est d'ouvrir des portes à d'éventuelles pistes d'améliorations qui pourraient servir de base à un prochain PIE.

Natural Language Understanding : Bien que l'utilisation de SnipsNLU soit satisfaisante, la base de données est un élément clé de son bon fonctionnement. Elle a pour l'instant été complétée pour les cas d'utilisations définis précédemment, et uniquement pour le vol Paris - Toulouse. Il pourrait être intéressant de compléter cette dernière, voire d'automatiser sa complétion selon l'endroit où se trouve l'avion, le type de vol etc. Un fonctionnement en ligne est aussi envisageable pour aller chercher les informations pertinentes (listes de tous les aéroports, des waypoints, etc.) sur des serveurs centralisés.

Interface Graphique : L'interface présente sur la maquette pour le moment est assez rustique. Faute de temps, aucune recherche n'a été réalisée pour adapter son apparence dans le cadre de son utilisation dans ce projet. Il pourrait être intéressant d'y consacrer du temps afin de mettre en place une réelle interface homme machine qui prend en compte l'environnement existant : le cockpit.

Ergonomie d'utilisation : Comme nous l'avons mentionné plus tôt, la maquette n'a pas fait l'objet d'études approfondies en ce qui concerne son utilisation concrète et son ergonomie dans un cockpit. Nous avons eu quelques retours d'utilisation mais rien de suffisant qui nous permettraient d'avoir une démarche scientifique aboutie en design d'interface. Afin d'avoir un produit réfléchi, il est nécessaire d'avoir cette phase avec des sujets qui vont permettre l'optimisation de ses fonctionnalités dans le cadre des interactions homme-machine. Il faudrait réaliser l'expériences avec un échantillon représentatif de sujets pour avoir des résultats quantifiés et objectifs. Plus tôt, nous avons proposé une réponse vocale intégrée au code qui mériterait une attention particulière dans le cadre de cette démarche future.

Commande vocale : Comme mentionné dans la section dédiée à la performance de la maquette, la commande vocale introduit une latence que nous ne pouvons pas contrôler. Elle est simplement due au fait que le programme doit comprendre que la phrase est finie. La solution la plus simple serait l'implémentation d'un bouton push-to-talk évoqué précédemment. Mais un meilleur entraînement de l'IA à comprendre la fin des requêtes pourrait améliorer la précision de détection.

Conclusion

En début d'année, nous avons récupéré le résultat du PIE précédent qui avait abouti sur un début de produit. Nous disposions donc d'une interface graphique dans laquelle l'utilisateur pouvait rentrer une requête rédigée d'une façon pré-définie au caractère près. Après ça, le programme nous retournait l'information voulue. L'information était récupérée dans une ontologie qui comportait trois grands groupes d'information : la navigation, la performance de l'avion et la météo.

Au cours de ce projet, nous avons travaillé sur l'amélioration de cette maquette à travers plusieurs points. La mission, convenue avec le client, était de rendre la maquette plus facilement utilisable et plus efficace. Pour ce faire, nous avons travaillé sur les points suivant :

— Programmation des requêtes en langage naturel ;

— Enrichissement de l'ontologie et des fonctionnalités de la maquette ;

— Intégration d'une entrée vocale et d'une sortie audio.

Les solutions choisies pour chacunes de ces améliorations sont les suivantes. Pour la programmation en langage naturel, c'est la bibliothèque SnipsNLU que nous avons retenue. Sa facilité d'utilisation et son évolutivité permettent d'utiliser un puissant outil de Natural Language Understanding au service de notre assistant virtuel. Les résultats sont satisfaisants sur les cas d'utilisation traités, mais pourraient être améliorés par une optimisation et un enrichissement de la base de données.
Pour ce qui est de l'enrichissement de l'ontologie, nous avons réalisé deux entretiens pilote. Ce sont deux pilotes de ligne A320 qui se sont prêté au jeu et qui nous ont aidé à trouver des points à améliorer par rapport aux fonctionnalités déjà existantes. Nous avons alors implémenté, en parallèle de ce qui avait été fait l'année dernière, des fonctions relatives aux checklists, au déroutements planifiés et au CPDLC. Enfin, le dernier développement que nous avons trouvé pertinent était le fait de pouvoir fonctionner oralement. Cela nous avait été suggéré par le client et par les pilotes lors de l'entretien. Pour la reconnaissance vocale, nous avons opté pour l'utilisation des librairies pythons dédiées à cela (PyAudio, Port Audio et SpeechRecognition). Pour la réponse orale, le module gTTS (Google Text-To-Speech) nous a permis de mettre en place un retour ergonomique. Cette librairie permet une grande flexibilité notamment au niveau des langues utilisables.

Parmi les objectifs que nous nous étions fixés en octobre 2021, tout n'a pas été réalisé. En effet, nous avions pensé au début à une phase de tests et d'expérience plus orientée vers les facteurs humains avec notre maquette. Ces points n'étaient pas réalisables au vu des ressources que nous avions alloués pour ce projet. Cependant, nous avons laissé la porte ouverte à de futurs évolutions. Nous avons pris le temps d'expliquer pourquoi nous avons implémenté certaines fonctionnalités et nous avons entamé une démarche itérative en prenant deux premiers feedback pilote. Ceci est repris dans la section "Pistes d'améliorations". Les points réguliers avec le client et l'organisation rigoureuse du projet nous ont permis d'avoir un produit final qui correspond aux attentes définies en conséquence.

Bibliographie

[1] AUTHIER L, CAUDIN E, MAITRE C, POTEL M. IA pour les opérations cockpits civils. ISAE SUPAERO industrial project. 2019.

[2] Commandes vocales Tesla ;. Available at https ://www.tesla.com/fr$_F R/support/voice-commands$.

[3] Airbus. Autonomous Flight ;. Available at https ://www.airbus.com/innovation/autonomous-and-connected/autonomous-flight.html.

[4] E A, L M. An Overview of Chatbot Technology. Artificial Intelligence Applications and Innovations : 16th IFIP WG 125 International Conference. 2020.

[5] Coucke A, Saade A, Ball A, Bluche T, Caulier A, Leroy D, et al. Snips Voice Platform : an embedded Spoken Language Understanding system for private-by-design voice interfaces. arXiv preprint arXiv :180510190. 2018 :12–16.

[6] Justicia Mayoral JA MarcLorente Miravalles. Prototyping of an artificial agent for an aircraftcockpit. ISAE SUPAERO Research project. 2020.

[7] Lemke U, Besser J. Cognitive load and listening effort : Concepts and age-related considerations. Ear and Hearing. 2016 ;37 :77S–84S.

Chapitre 5

Annexes

```yaml
# Airport Entity
---
type: entity
name: airport
values:
  - [LFPO, Paris Orly, Paris, Orly]
  - [LFBO, Toulouse, Toulouse Blagnac, Blagnac]
  - [arrival, destination]
  - departure
```

Fig. 5.1 – Exemple d'entité implémentée en YAML

```yaml
# weatherAirport Intent
---
type: intent
name: weatherAirport
slots:
  - name: place
    entity: airport
  - name: info
    entity: weatherInfo
utterances:
  - give me the [info] at [place]
  - i need the [info] in [place]
  - what is the [info] at [place] ?
  - can i have the [info] at [place] ?
```

Fig. 5.2 – Exemple de requête implémentée en YAML

Fig. 5.3 – Diagramme de Gantt prévu au 05/12/2020

Fig. 5.4 – Diagramme de Gantt effectif au 24/03/2021