



Master Sciences et Techniques
Département de Physique
Filière : Systèmes embarqués et robotique
Projet de fin d'études

Développement d'une application pour le filtrage des Options fonctionnelles et correspondance des projets pour BSI

(expleo)

Projet réalisé par :

✓ BOUTAIBI Nouhaila

Encadré par :

- ✓ M. EL FATHI Amine
- ✓ M. BEN TAHER
Mohamed Ayoub
(Expleo)

Présenté devant les membres du jury composé de :

- Mme. ZKHNINI Samira, Présidente
- M. EL AOUADI Ibrahim, Examinateur
- M. EL FATHI Amine, Encadrant

Année Universitaire : 2024/2025

REMERCIEMENTS

Avant d'entamer, je tiens à remercier **Allah**, qui m'a facilité le chemin pour réaliser ce projet et m'a donné le courage ainsi que la patience nécessaires pour surmonter les obstacles, les échecs et les moments de découragement.

J'adresse également mes sincères remerciements à mes deux encadrants, **professionnel et académique** : M. **Mohamed Ayoub BENTAHER**, au sein d'**Expleo Maroc**, et M. **Amine EL FATHI**, au sein de la **FSTH**, pour leur engagement indéfectible, leur soutien inestimable et leurs conseils éclairés tout au long de cette période de stage.

Je remercie aussi mes **team leaders**, M. **Soufiane CHAKRI**, M. **Ali BENHSI-NAT** et M. **Elmehdi HEBROUNE**, pour la confiance qu'ils m'ont accordée afin de mener ce projet à bien.

Je souhaite exprimer toute ma reconnaissance à l'ensemble des **enseignants** ayant contribué à mon parcours académique, en particulier **mes professeurs de la FSTH**.

Je tiens également à remercier les **membres du jury** pour avoir accepté d'évaluer mon travail et pour l'avoir enrichi de leurs remarques précieuses.

Un grand merci à ma **petite famille** pour son soutien, tant moral que financier, durant toutes ces années de sacrifices et de patience.

Enfin, il me semble indispensable de mentionner toute personne ayant, de près ou de loin, contribué à la réalisation de ce projet.

DÉDICACE

À ma merveilleuse mère,

Tu es la source de ma vie, mon guide, ma confidente. Dans les hauts comme dans les bas, tu étais là pour moi, m'encourageant et me soutenant inconditionnellement.

À mon cher père,

Bien que tu ne sois plus physiquement présent, ton esprit continue de m'inspirer chaque jour. Tu étais un modèle de force, de sagesse et de bienveillance. Je suis honorée d'être ta fille, de porter avec moi les principes que tu m'as inculqués.

À mes chers frères,

Vous avez été mes piliers, mes sources constantes de motivation. Votre présence et vos paroles inspirantes ont nourri mon esprit et allumé en moi une flamme d'audace et de détermination.

À ma chère nièce,

Tu es un rayon de soleil dans nos vies, apportant la joie et le bonheur à notre famille. Ton innocence et ta pureté d'esprit sont des atouts précieux me permettront de voir la beauté et la magie dans les petites choses de la vie.

RÉSUMÉ

Ce projet s'inscrit dans une démarche d'automatisation du traitement des exigences techniques liées aux projets clients. Dans les processus industriels, l'analyse manuelle de ces exigences représente une tâche chronophage et sujette à erreurs humaines. C'est pourquoi ce travail propose le développement d'une application logicielle capable d'extraire automatiquement les exigences à partir de documents structurés, d'en analyser la logique interne et de vérifier leur applicabilité par rapport aux caractéristiques des projets en cours.

L'application proposée a été développée en Python, en utilisant le framework PyQt5 pour l'interface graphique et SQLite comme système de gestion de base de données, afin de garantir une solution efficace, intuitive et évolutive.

L'application fonctionne efficacement sur des documents structurés. Toutefois, certaines limitations subsistent, notamment la diversité des formats de fichiers d'exigences, qui rend difficile une prise en charge totalement universelle.

Mots-clés : automatisation, exigences, projets, applicabilité, Python, PyQt5, SQLite

ABSTRACT

This project is part of an approach to automate the processing of technical requirements related to client projects. In industrial processes, manual analysis of these requirements is a time-consuming task prone to human error. This work therefore proposes the development of a software application capable of automatically extracting requirements from structured documents, analyzing their internal logic, and verifying their applicability according to the characteristics of ongoing projects.

The proposed application was developed in Python, using the PyQt5 framework for the graphical user interface and SQLite as the database management system, in order to provide an efficient, intuitive, and scalable solution.

The application performs effectively on structured documents. However, some limitations remain, particularly the diversity of requirement file formats, which makes fully universal support challenging.

Keywords : automation, requirements, projects, applicability, Python, PyQt5, SQLite

TABLE DES MATIÈRES

Remerciements	i
Dédicace	ii
Résumé	iii
Abstract	iv
LISTE DES ABRÉVIATIONS	xii
Introduction générale	1
1 Contexte général du projet	2
1.1 Introduction	2
1.2 Présentation de EXPLEO GROUP	2
1.2.1 Historique	2
1.2.2 Organisation	3
1.2.2.1 Fiche d'identification	3
1.2.2.2 Organigramme	3
1.2.3 Département d'électronique et système embarqué – Expleo Maroc	4
1.2.3.1 AEE (Architecture Electrique Electronique)	4
1.2.3.2 HIL Validation	5
1.2.3.3 Test et validation de Powertrain, Châssis, ADAS	6
1.2.3.4 Conception des composants de Powertrain, Châssis et ADAS	6

1.2.3.5	Conception fonctionnelle pour l'éclairage extérieur et la signalisation	7
1.3	Client principal : Groupe STELLANTIS	7
1.4	Cadrage de projet	8
1.4.1	Validation du calculateur BSI	8
1.4.1.1	BSI	9
1.4.1.2	Validation fonctionnelle	9
1.4.1.3	Génération de campagne	10
1.4.1.4	Validation	16
1.5	Présentation du projet	18
1.6	Contexte du projet	19
1.7	Problématique	20
1.8	Objectifs	21
1.9	Solution proposée	21
1.10	Besoins fonctionnels	22
1.11	Besoins non fonctionnels	22
1.12	Contraintes techniques	23
1.13	Déroulement du projet	23
1.13.1	Tableau des tâches	23
1.13.2	Clarification des tâches	23
1.13.3	Méthodologie	24
1.14	Conclusion	25
2	État de l'art des systèmes automobiles	26
2.1	Introduction	26
2.2	Les types de Véhicules	26
2.2.1	HEV	27
2.2.2	MHEV	27
2.2.3	PHEV	28
2.2.4	BEV	28
2.2.5	FCEV	28
2.3	Architecture Electrique Electronique	28
2.3.1	Les calculateurs automobiles	29
2.3.2	Bus de communication	30
2.3.3	Protocole LIN	30
2.3.3.1	Format des trames LIN	31
2.3.3.2	Avantages et Limitations de LIN	32
2.3.4	Le LIN dans les architectures Stellantis	32

2.3.5	Protocole CAN	33
2.3.5.1	Les signaux du bus CAN	34
2.3.5.2	La longueur des bus CAN	35
2.3.5.3	Le Diagnostic	35
2.3.6	L'évolution du nombre de calculateur chez Stellantis	37
2.3.7	L'évolution des architectures chez Stellantis	37
2.3.8	Exemple d'une architecture AEE2010	39
2.3.9	BSI	40
2.4	Banc et environnement de test HIL	41
2.4.1	Test HIL	41
2.4.1.1	définition et généralités	41
2.5	Conclusion	44
3	Conception et réalisation de l'application	45
3.1	Introduction	45
3.2	Conception de l'application	45
3.2.1	Langage de modélisation UML	45
3.2.2	Diagramme de cas d'utilisation	46
3.2.3	Diagramme de classe	48
3.2.4	Diagramme des séquences	50
3.2.4.1	Diagramme de séquence d'insertion des exigences	51
3.2.4.2	Diagramme de séquence de comparaison des exigences	52
3.2.4.3	Diagramme de séquence de la liste des exigences	53
3.2.4.4	Diagramme de séquence de suppression d'une exigence	54
3.2.4.5	Diagramme de séquence de vérification d'applicabilité avec projets	55
3.2.4.6	Diagramme de séquence de Modification de la base de données	56
3.2.4.7	Diagramme de Confirmation administrateur avant suppression	58
3.3	Conception de la base de données	59
3.3.1	Outils utilisés	60
3.3.1.1	Langage principal Python 3	60
3.3.1.2	Environnement de développement VS Code	62
3.3.1.3	Framework PyQt5	62
3.3.1.4	Personnalisation de l'UI avec QSS	63
3.3.1.5	Manipulation de fichiers Word avec python-docx	63
3.3.1.6	Manipulation de fichiers Excel avec pandas	64

3.3.1.7	SQLite	64
3.3.1.8	Protocole SMTP	65
3.3.1.9	PyInstaller	65
3.4	Réalisation de l'application	65
3.4.1	Menu Principal	65
3.4.2	Interface d'insertion des exigences	66
3.4.3	Interface de comparaison des exigences	67
3.4.4	Interface de la liste des exigences	68
3.4.5	Interface Comparaison avec projets	69
3.4.6	Interface de Modification de la base de données	71
3.4.7	Interface de Gestion des Projets	71
3.4.8	Interface de Suppression des projets	72
3.5	Conclusion	74
	Conclusion générale	75

TABLE DES FIGURES

1.1	Organigramme ESE Expleo Group	4
1.2	Diagramme du cycle en V	5
1.3	Groupe STELLANTIS	8
1.4	Exemple du calculateur BSI	9
1.5	Exemple de fichier DecliEE	12
1.6	Exemple de fichier de Silhouette	13
1.7	Diagramme de processus de génération de campagne	14
1.8	Extrait de document de campagne	15
1.9	Diagramme de vérification d'applicabilité d'une thématique	15
1.10	Première forme des exigences d'une fonction X	17
1.11	Deuxième forme des exigences d'une fonction Y	17
1.12	Exemple de fichier de plan de test	18
1.13	Processus de validation de BSI	20
1.14	Processus Agile	25
2.1	Définitions et types d'architecture de véhicule électrique	27
2.2	Quelques calculateurs Automobiles	29
2.3	Format des trames LIN	31
2.4	Le LIN dans les architectures Stellantis	33
2.5	Réseau CAN Automobile	33
2.6	CAN High Speed	34
2.7	CAN Low Speed	35
2.8	La relation Débit/Longueur dans les bus CAN	35
2.9	L'évolution du nombre de calculateur	37
2.10	Les générations des architectures chez Stellantis	38

2.11	Exemple d'une architecture AEE2010	39
2.12	Calculateur BSI et son emplacement dans les véhicules	40
2.13	Principe de Hardware in the loop	42
2.14	Environnement du test du BSI	42
2.15	BSI sous test	43
2.16	Banc de test HIL	43
3.1	Logo d'UML	46
3.2	Diagramme de cas d'utilisation	47
3.3	Diagramme de classe de l'application	50
3.4	Diagramme de séquence d'insertion des exigences	51
3.5	Diagramme de séquence de comparaison des exigences	52
3.6	Diagramme de séquence la liste des exigences	53
3.7	Diagramme de séquence de suppression d'une exigence	54
3.8	Diagramme de séquence de vérification d'applicabilité avec projets	55
3.9	Diagramme de séquence de Modification de la base de données	56
3.10	Diagramme de Confirmation administrateur avant suppression	58
3.11	Table des exigences	59
3.12	Table des projets	60
3.13	Logo de Python	61
3.14	Logo de Anaconda et VS Code	62
3.15	Logo de PyQt	63
3.16	python-docx	63
3.17	Logo de pandas	64
3.18	Logo de SQLite	65
3.19	Menu Principal de l'application	66
3.20	Interface d'insertion des exigences	67
3.21	Interface de comparaison des exigences	68
3.22	Interface de la liste des exigences	69
3.23	Interface de Résultat par projet	70
3.24	Interface de Résumé global	70
3.25	Interface de Modification de la base de données	71
3.26	Interface de Gestion des Projets	72
3.27	Interface de Suppression des projets	73
3.28	Confirmation d'admin avant suppression	73

LISTE DES TABLEAUX

1.1	Fiche d'identification de la société Expleo Group	3
1.2	Méthode QQOQCP	21
1.3	Tableau des tâches	23
3.1	Comparaison des langages de programmation : Python, VBA et C . . .	61

LISTE DES ABRÉVIATIONS

- BSI** : Boîtier de Servitude Intelligent
ESE : Électrique et Système Embarqué
ECU : Electronic Control Unit
AEE : Architecture Electrique Electronique
DecliEE : Déclinaison Électrique Électronique
HW : Hardware
SW : Software
HEV : Hybrid Electric Vehicle
MHEV : Mild Hybrid Electric Vehicle
PHEV : Plug-in Hybrid Electric Vehicle
BEV : Battery Electric Vehicle
FCEV : Fuel Cell Electric Vehicle
ADAS : Advanced Driver Assistance Systems
IHM : Interface Homme-Machine
ECM : Engine Control Module
PCM : Powertrain Control Module
BCM : Brake Control Module
LIN : Local Interconnect Network
Sync : Synchronisation
ID : Identifiant
CAN : Controller Area Network
CAN HS : CAN High Speed
CAN LS : CAN Low Speed
CAN I/S : CAN inter-système
CAN LAS : CAN liaison au sol
CAN FD : CAN Flexible Data-Rate
MOST : Media Oriented Systems Transport

ODX : Open Diagnostic data eXchange
VSM : Vehicle Signal Master
AVAS : Acoustic Vehicle Alerting System
PSA : Peugeot Société Anonyme
FCA : Fiat Chrysler Automobiles
ISO : International Organization for Standardization
ABS : Anti-lock Braking System
UDS : Unified Diagnostic Services
KWP2000 : Keyword Protocol 2000
DTC : Diagnostic Trouble Code
DID : Data Identifier
DoIP : Diagnostics over IP
DoCAN : Diagnostics over CAN
OTA : Over-The-Air
TLS : Transport Layer Security
GMP : Groupe Moto-Propulseur
InfoDiv : infodivertissement
CAR : Carrosserie
CONF : Confort
IHM : interface homme-machine
VAN : Vehicle Area Network
NEA : New Electronic Architecture
SEV : Système Électrique Véhicule
RCD : Réveil Commandé à Distance
JDD : Journal des Défauts
TCP/IP : Transmission Control Protocol/Internet Protocol
SGBD : Système de Gestion de Base de Données
UML : Unified Modeling Language
NA : Non Appliquée
VS Code : Visual Studio Code
QSS : Qt Style Sheets
SQL : Structured Query Language
SMTP : Simple Mail Transfer Protocol
HIL : Hardware In the Loop
UI : User Interface
NLP : Natural Language Processing

INTRODUCTION GÉNÉRALE

DANS le secteur automobile en constante évolution, l'intégration croissante de fonctions électroniques et logicielles au sein des véhicules modernes impose des exigences toujours plus élevées en matière de validation, de fiabilité et d'automatisation. Les systèmes embarqués, en particulier ceux pilotés par des calculateurs comme le BSI (Boîtier de Servitude Intelligent), jouent un rôle central dans la coordination des fonctionnalités du véhicule, allant de la signalisation au confort, en passant par la sécurité.

Au sein d'Expleo Group, une entreprise d'ingénierie spécialisée dans l'innovation technologique et son client stratégique le groupe Stellantis, les activités de validation de ces systèmes nécessitent une rigueur accrue, notamment dans la gestion des exigences fonctionnelles liées aux projets. La vérification de l'applicabilité de ces exigences aux différents modèles de véhicules représente un processus critique, souvent long, complexe et sujet à erreurs lorsqu'il est réalisé manuellement.

C'est dans ce contexte qu'intervient le présent travail, qui vise à concevoir et développer une application logicielle destinée à automatiser le traitement des exigences, à améliorer leur comparaison, et à vérifier leur compatibilité avec divers projets. Ce projet s'inscrit dans une logique d'optimisation des processus de validation de calculateur BSI, en apportant un outil fiable, traçable et évolutif.

Ce rapport est structuré en trois chapitres étroitement liés. Le premier chapitre établit le cadre général du projet, en présentant le contexte industriel, les besoins identifiés, et en mettant en évidence les enjeux de la validation du calculateur BSI. Le deuxième chapitre approfondit les notions techniques nécessaires à la compréhension du projet en dressant l'état de l'art des systèmes automobiles, avec un accent particulier sur les types de véhicules, l'architecture EE, le calculateur BSI et le banc de test HIL, des éléments directement liés aux processus de validation du BSI. Enfin, le troisième chapitre détaille la conception, le développement et la mise en œuvre de l'application répondant à la problématique.

CHAPITRE 1

CONTEXTE GÉNÉRAL DU PROJET

1.1 Introduction

Ce chapitre se focalise sur la présentation de l'entité d'accueil EXPLEO GROUP, ainsi que sur le contexte du projet réalisé durant notre stage au sein de cette société. Nous débuterons par une présentation de l'entreprise, en mettant en lumière son parcours historique, sa raison d'être, son organisation interne et sa place dans l'industrie. Ensuite, nous aborderons le cadre spécifique du projet, en exposant les motivations ayant conduit à sa mise en œuvre, les défis identifiés, ainsi que les objectifs fixés.

1.2 Présentation de EXPLEO GROUP

1.2.1 Historique

La fusion d'Assystem Technologies et de SQS en 2017 a conduit à la création d'Expleo Group. Il propose des prestations de consultation, d'ingénierie et de qualité aux domaines de l'aérospatiale, du transport, de l'énergie, des télécommunications et des services financiers.

Depuis ses débuts, Expleo Group a toujours été dévoué à proposer des solutions de transformation numérique de grande qualité à ses clients. La société a continué son expansion géographique en établissant de nouveaux bureaux dans divers pays, y compris l'Inde, la Chine et l'Australie.

Expleo Group intervient également dans le secteur automobile, offrant une variété de

services de consultation, d'ingénierie et de qualité aux fabricants automobiles, équipementiers et fournisseurs de systèmes pour véhicules.

Expleo Group propose divers services dans le secteur automobile, y compris la conception et l'élaboration de systèmes de propulsion électrique et hybride, le développement de logiciels intégrés pour les véhicules, ainsi que la validation et la vérification.

Expleo Group offre un éventail de services dans le secteur automobile, englobant la conception et l'élaboration de systèmes de propulsion électrique et hybride, le développement de logiciels embarqués aux véhicules, la validation et la vérification des systèmes automobiles, sans oublier la gestion qualité des produits et des processus.

En 2019, Expleo Group a établi un partenariat stratégique avec Renault-Nissan-Mitsubishi, l'un des plus grands groupes automobiles à l'échelle mondiale, en vue de fournir des services consultatifs en ingénierie et en qualité à travers le monde. Ce partenariat reflète la compétence d'Expleo Group dans le secteur automobile et son dévouement à collaborer avec les acteurs majeurs de l'industrie pour définir le futur de l'automobile [1].

1.2.2 Organisation

1.2.2.1 Fiche d'identification

Voici les informations d'identification de l'entreprise Expleo Group présentées sur le tableau ci-dessous :

La raison sociale	Expleo Group
Année de création	2017(fusion d'Assystem Technologies et de SQS)
Président-directeur général	Rajesh Krihnamurthy
La forme juridique	Société Anonyme Simplifiée
Nombre d'employés	Plus de 600
Chiffre d'affaires	1,3 Milliard de Dollar
Le siège social	La Garenne-Colombes, France
Le site web	https://expleo.com/global/en/

TABLE 1.1 – Fiche d'identification de la société Expleo Group

1.2.2.2 Organigramme

L'organigramme du département ESE (Électrique et Système Embarqué) d'Expleo Group, section marocaine, est illustré ci-dessous.

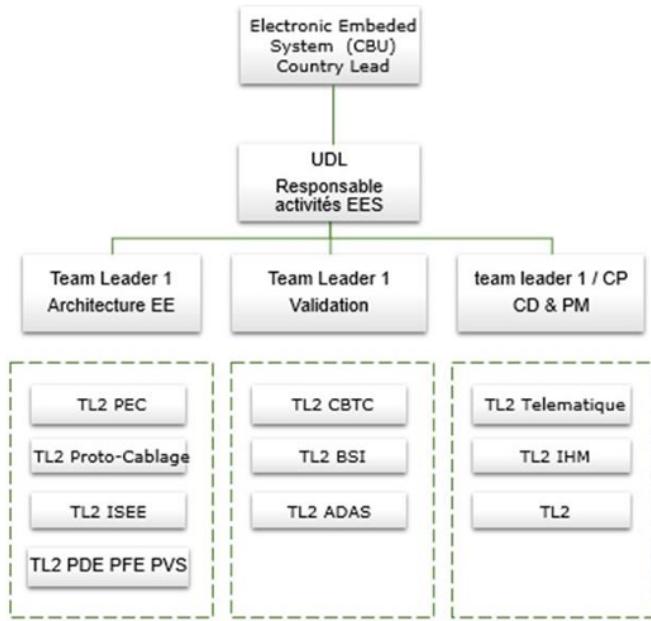


FIGURE 1.1 – Organigramme ESE Expleo Group

1.2.3 Département d'électronique et système embarqué – Expleo Maroc

Il est essentiel de présenter le département ESE et les diverses activités qu'il comprend pour définir le contexte de notre projet. Dans ce département, sont mis en œuvre divers processus pour satisfaire les exigences et spécifications techniques demandées par nos clients dans une multitude de secteurs. Ces actions sont menées pour garantir le contentement des clients et assurer la qualité et la fiabilité des produits que nous concevons.

1.2.3.1 AEE (Architecture Electrique Electronique)

Expleo Group offre à Stellantis, son client privilégié dans le secteur des systèmes embarqués, une vaste sélection de services. On propose les services suivants :

- La définition technique et l'analyse fonctionnelle des systèmes HW (Hardware) & SW (Software)
- L'évaluation de l'impact électrique
- L'analyse géométrique
- La fourniture et la disponibilité des composants électriques

- La modélisation de harnais en 3D/2D
- La création physique de harnais et la validation de leurs conformités
- Les tests d'assemblage, de lancement et la validation fonctionnelle, ainsi que la fourniture du harnais du véhicule d'essai

En collaborant de manière étroite avec les équipes de Stellantis, Expleo Group assure la fiabilité et l'excellence des harnais. Les procédures d'Expleo Group sont flexibles et ajustables pour s'adapter aux exigences particulières de chaque projet, tout en maintenant des standards de qualité rigoureux.

1.2.3.2 HIL Validation

La figure ci-dessous illustre un modèle de développement logiciel en cycle en V, qui décrit les diverses phases du processus de développement, allant de la définition des exigences à la validation finale. Il insiste sur la vérification et la validation à chaque phase pour assurer la qualité du système embarqué. Le département responsable de la validation du logiciel du véhicule met en œuvre la validation HIL (Hardware In the Loop).

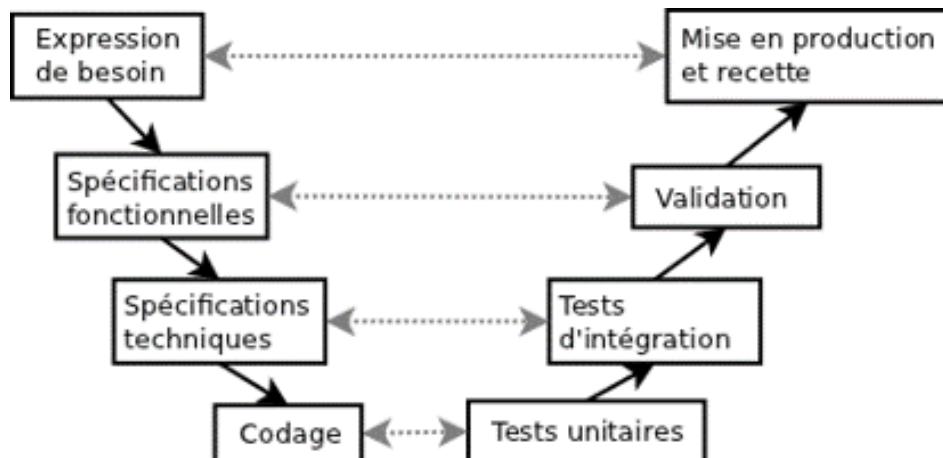


FIGURE 1.2 – Diagramme du cycle en V

La validation HIL implique l'intégration du système en cours de développement dans un environnement d'essai qui fusionne le matériel concret et le logiciel simulé.

C'est une validation effectuée à l'aide de simulateurs reliés au logiciel actuellement en développement. La validation est une procédure visant à garantir que le logiciel conçu respecte bien les spécifications et les demandes clairement établies par le client. Dans le contexte des systèmes embarqués, la validation constitue une phase cruciale du

cycle en V. Elle offre également l'assurance de la fiabilité, de l'efficacité et de la sécurité du logiciel, tout en garantissant sa conformité à toutes les spécifications établies.

Pour résumer, voici les buts de cette équipe :

- Assurer la validation fonctionnelle de 80 fonctionnalités distinctes comme la climatisation, le contrôle de vitesse, l'éclairage, etc.
- Mettre en œuvre la validation HIL pour les programmes en étape de développement et lors de la production en série.

Ces objectifs sont réalisés par le biais de différentes phases séquentielles, chaque étape ayant pour but de contrôler et d'approuver les données entrantes par rapport aux exigences et spécifications définis par le client, puis transférer les données traitées à l'étape suivante.

1.2.3.3 Test et validation de Powertrain, Châssis, ADAS

Cette tâche consiste à valider les softwares pour divers ECU (Electronic Control Unit) dans le secteur automobile, comme le moteur, la batterie, le chargeur embarqué, etc. Elle porte sur les véhicules à moteur thermique, hybride et électrique. Parmi les activités offertes, on retrouve :

- L'établissement de plans de test.
- Le développement de cas de test.
- L'exécution de tests manuels et automatisés.
- L'analyse des anomalies et le réglage des valeurs d'étalonnage.

Cela garantit la conformité et la performance des systèmes électroniques des voitures.

1.2.3.4 Conception des composants de Powertrain, Châssis et ADAS

Cette activité développe des softwares pour différents ECUs tels que Gearbox, circuit breaker. Elle concerne le Powertrain des voitures hybrides, hybrides rechargeables et électriques, ainsi que le châssis et les ADAS (Advanced Driver Assistance Systems). Les équipes de cette fonction ont pour responsabilités :

- Spécifier et développer des softwares en identifiant toutes les spécifications applicables au projet et en synthétisant les configurations pour divers projets.
- Créer des applications de messagerie pour chaque calculateur et organe du véhicule, y compris les protocoles de communication CAN (Controller Area Network), CANFD (CAN Flexible Data-Rate), LIN (Local Interconnect Network), Flexray, etc, ainsi que la base de données CAN.
- Élaborer des spécifications techniques pour le développement, l'usine, le service après-vente et les fournisseurs, y compris les besoins en matière de diagnostic.

- Créer des diagnostics de messagerie et des services de diagnostic pour générer le fichier ODX (Open Diagnostic data eXchange), ainsi que des matrices de diagnostic. Cette activité joue un rôle clé dans le développement et la maintenance des systèmes électroniques des véhicules.

1.2.3.5 Conception fonctionnelle pour l'éclairage extérieur et la signalisation

L'objectif de cette activité est d'identifier les spécifications logicielles de différents systèmes électroniques embarqués, tels que VSM (Vehicle Signal Master), AVAS (Acoustic Vehicle Alerting System), etc, dans le cadre de l'éclairage et de la signalisation extérieurs. Pour ce faire, plusieurs services ont été mis en place :

- Les exigences fonctionnelles des logiciels comprennent l'identification de toutes les spécifications pertinentes du projet, la synthèse des configurations de diversité fonctionnelle pour différents projets, l'élaboration d'une attribution des besoins fonctionnels et de diagnostics, et la garantie de la traçabilité des exigences.
- Architecture fonctionnelle : développer des fonctions techniques, des flux efficaces et des vues fonctionnelles correspondantes.
- Analyse d'impact : déterminer comment les nouveaux développements affecteront les préoccupations opérationnelles, les questions de sécurité et les contraintes réglementaires ; analyser les problèmes techniques et proposer des solutions.
- Gestion : aligner la conception sur la qualité, le coût, la performance et la planification.

1.3 Client principal : Groupe STELLANTIS

Le groupe STELLANTIS est le principal client auquel l'entreprise fournit ses services. Il s'agit d'une multinationale de l'automobile fondée le 16 janvier 2021 à la suite de la fusion du groupe PSA (Peugeot Société Anonyme) et de FCA (Fiat Chrysler Automobiles).



FIGURE 1.3 – Groupe STELLANTIS

Le groupe STELLANTIS exploite et commercialise quatorze marques automobiles, dont 7 appartiennent au groupe PSA (Citroën, DS Automobiles, Opel, Peugeot, Vauxhall, Free2move et Leasys) et 9 à FCA (Abarth, Alfa Romeo, Chrysler, Dodge, Fiat, Jeep, Lancia, Maserati et RAM).

1.4 Cadrage de projet

Dans cette partie, nous allons exposer le contexte de notre projet, après avoir détaillé les actions de l'équipe HIL validation concernant le calculateur BSI. Où l'on expose minutieusement toutes les tâches effectuées lors de la validation afin d'identifier et de préciser les aspects cruciaux de cette procédure.

1.4.1 Validation du calculateur BSI

L'équipe de validation HIL est chargée de valider les fonctionnalités de nouveaux softwares en se basant sur le calculateur BSI via des simulateurs reliés au logiciel. La validation est effectuée conformément à la norme ISO 26262 (International Organization for Standardization), qui représente une approche globale visant à garantir la sécurité des systèmes électroniques des véhicules pour protéger les conducteurs contre les défaillances électroniques et de software.

1.4.1.1 BSI

Le BSI est un dispositif électronique intégré dans les véhicules automobiles qui joue un rôle crucial dans la centralisation et le contrôle des fonctions électriques et électroniques du véhicule. En tant que "cerveau" de la voiture, il permet la gestion simplifiée et la coordination efficace des sous-systèmes électriques. La figure ci-dessous représente un modèle du calculateur BSI :



FIGURE 1.4 – Exemple du calculateur BSI

1.4.1.2 Validation fonctionnelle

Dans le département, l'équipe de validation assume une fonction cruciale en menant une validation fonctionnelle détaillée. Cette validation fonctionnelle implique la mise en œuvre de tests en utilisant divers scénarios pour contrôler si les résultats obtenus respectent les exigences et les spécifications définies. Il est important de préciser que l'approche de validation adoptée est la validation HIL. Dans ce cadre, le matériel physique est simulé à l'aide de simulateurs électroniques, alors que le logiciel de contrôle est évalué dans un environnement simulé qui imite les conditions d'exploitation réelles du système. Cette technique de validation offre la possibilité de contrôler le fonctionnement du système dans des conditions extrêmes ou compliquées à reproduire en se servant du matériel physique réel.

La validation HIL aide à identifier les problèmes de compatibilité matérielle et logicielle, les défauts de conception ainsi que les bugs logiciels avant l'implémentation du système sur terrain.

Elle permet aussi d'optimiser les tests tout en diminuant les frais de développement en évitant les essais sur l'équipement physique réel, qui peuvent s'avérer coûteux et demander beaucoup de temps.

Le déclenchement de la validation au sein de l'organisme se fait après la réception d'une demande de validation d'une fonction pour un nouveau software de la part du client.

Dans Expleo, l'équipe de validation HIL est responsable de la validation des nouveaux softwares pour des fonctions spécifiques.

La validation des fonctions pour différents softwares est une étape critique dans la conception de modèles de voiture. L'idée est de généraliser les fonctions et de tester leur compatibilité avec une variété de projets (nouveaux softwares), afin d'assurer leur robustesse et leur fiabilité dans divers contextes et situations. Cette méthode permet de s'assurer que les fonctions sont suffisamment robustes et qu'elles peuvent être utilisées de manière fiable sur un large éventail de projets.

Dans un véhicule automobile, les fonctions se rassemblent en différentes catégories, telles que les fonctions de sécurité, de confort et d'assistance à la conduite, etc. ces fonctions qui se composent de plusieurs options (thématiques). Prenant par exemple, une fonction de sécurité peut comprendre des options telles que l'airbag, l'ABS (Anti-lock Braking System) ou la détection de collision. Ces options sont regroupées dans la même catégorie car elles ont toutes pour objectif de garantir la sécurité des passagers et des conducteurs. En regroupant les options de cette manière, les ingénieurs automobiles peuvent concevoir et développer des fonctions de manière plus efficace, en se concentrant sur les besoins et les préférences des clients.

Prenant par exemple la fonction **PICC** (Prioriser les informations de conduite et confort), elle fait partie des fonctions de confort et d'assistance. Cette fonction rassemble toute les thématiques (options) du véhicule qui sont liées à la signalisation et l'affichage de n'importe quelle donnée qui sert à informer le conducteur de son état de confort ou de contrôle du véhicule.

1.4.1.3 Génération de campagne

Suite à la réception d'une demande de validation pour une fonction, nous procédons d'abord à la création de la campagne relative à cette fonction que nous souhaitons tester. Ce processus se termine par l'envoi d'un livrable signalant toutes incohérences entre les divers documents d'entrée.

La génération de campagne sert à choisir les bons plans de test pour le projet, cette génération s'effectue après la réception d'une demande contenant :

La fonction à tester associé par son référentiel pour l'apport d'un fichier de test contenant l'ensemble des plans de test et les scénarios correspondant et un autre fichier comportant le résultat attendu de test.

Pour une fonction, souvent, nous trouverons plusieurs projets. Ces projets selon la demande peuvent se considérer comme des complémentaires au premier projet, ou peuvent se considérer comme des projets indépendants l'un de l'autre ou un mélange de ces deux cas.

NB : Chaque software est associé à un projet, d'où le projet représente un modèle de voiture.

Pour chaque software et son projet mentionné sur la demande de fonction deux fichiers seront importés :

- **DecliEE (Déclinaison Electrique Electronique)** : c'est un document d'entrée qui est consulté à la phase de mise à jour des plans de test, il comporte toute information dont on a besoin à propos de toutes les types de voiture liée à notre projet (électrique, thermique, hybride...), prenant en considération que chaque type peut avoir plusieurs configurations qui correspondent à des différentes spécifications (Type de boîte à vitesse, type de contact, type du boîtier télématique, type d'architecture électrique électronique...). Ce document mentionne l'applicabilité de toutes les thématiques (options) au niveau de toutes les configurations d'un modèle de voiture (c'est un document primordial dans le contexte de notre projet).

La figure ci-dessous représente un exemple de fichier DecliEE :

1.4. CADRAGE DE PROJET

FIGURE 1.5 – Exemple de fichier DeclEE

Cette figure ci-dessus représente la valeur d'applicabilité de quelques options sur le DecliEE pour plusieurs configurations :

- S : Applicable
 - O : Optionnel
 - Case vide : Non applicable

— **Silhouette** : c'est aussi un document que l'ingénieur doit apporter et qui fait partie des fichiers qui sont intégrés au software pour l'exécution de validation, il est particulier par rapport au DecliEE, d'où les configurations concernées sur le document dépend seulement de type de voiture de notre projet, il rassemble seulement les modèles des voitures de même type. La différence se manifeste aussi dans l'applicabilité des thématiques qui inclue seulement les thématiques de mêmes types des voitures.

La figure ci-dessous représente un exemple de fichier Silhouette :

1.4. CADRAGE DE PROJET

6	Nom de l'ODD	Valeur	TIT	Regroupement	Option Intérêse	Ne pas supprimer la colonne	Nom fonctionnel	Thématique	Nom STD	Comment	Valeur STD	Valeur STD GÉNÉRIQUE	C42E_MHEV_vDCT_M_DAG	C42E_MHEV_vDCT_M_DAG_Auto
7	BAA_00	sans	--	A4					COMPT	--		Xopt	X	X
8	BAA_01	sans	--	A4CVI_02					COMPT	--	--	--	--	
9	BAE_01	Z309	01	A ALARME	X				ALARFM	TYPE_A	Indique le type d'alarme (inclus alarme d'activation et de désactivation en Suisse)	opt (AVE)	Xopt	X
10	BAE_02	Z309	02	0 ALARME	X				ALARFM	TYPE_A	Indique le type d'alarme	opt (AVE)	--	--

FIGURE 1.6 – Exemple de fichier de Silhouette

Cette figure ci-dessus représente la valeur d'applicabilité de quelques thématiques sur la silhouette par rapport à deux configurations du projet :

X : applicable

– : Non applicable

La génération de campagne est une phase cruciale lors de la validation, car elle permet de préparer et de documenter les fichiers des tests nécessaires pour s'assurer que le software testé est conforme aux exigences fixées et qu'il est fiable, performant et sécurisé. Cette génération consiste à créer un document qui énumère les tests à effectuer et inclut les configurations fournies par le client. Chaque configuration est un nouveau software pour un projet défini. Afin de valider le software du projet, chaque configuration comprend des plans de test à valider et des méthodologies de test à suivre. Le plan de test de validation spécifie les tests qui seront effectués, les environnements de test qui seront utilisés et les procédures de test qui seront utilisées, les critères de réussite de test, les exigences de traçabilité, etc.

La génération de compagnie s'effectue par l'exécution d'une macro créée par le client (Stellantis), cette macro comporte des différents documents qui sont téléchargés à partir de site de Stellantis, ces documents sont :

- Le fichier de matrice de test
- Le fichier de résultat de test
- Le fichier de silhouette de projet
- Le fichier de DecliEE de projet

Pour le cas de la validation d'une fonction pour plusieurs projets, le choix de l'ajout d'une configuration dans la macro de compagnie est disponible.

Comme mentionné précédemment, il est courant de tester une fonction sur plusieurs projets différents en même validation, qu'ils soient considérés comme complémentaires ou indépendants les uns des autres. Pour le cas de projets complémentaires, une particularité se manifeste lors de la phase de validation des fichiers de test. En effet, Si un fichier de test est validé dans le projet principal ou le projet complémentaire, il ne sera pas systématiquement retesté. Cela permet de gagner du temps et de l'efficacité dans la validation de la fonction pour différents projets. En revanche, pour les projets indépendants, chaque fichier de test doit être validé pour chaque projet séparément, afin de garantir la compatibilité de la fonction avec chaque projet individuellement.

La figure ci-dessous est un diagramme qui représente l'enchaînement de processus pour la génération de campagne.

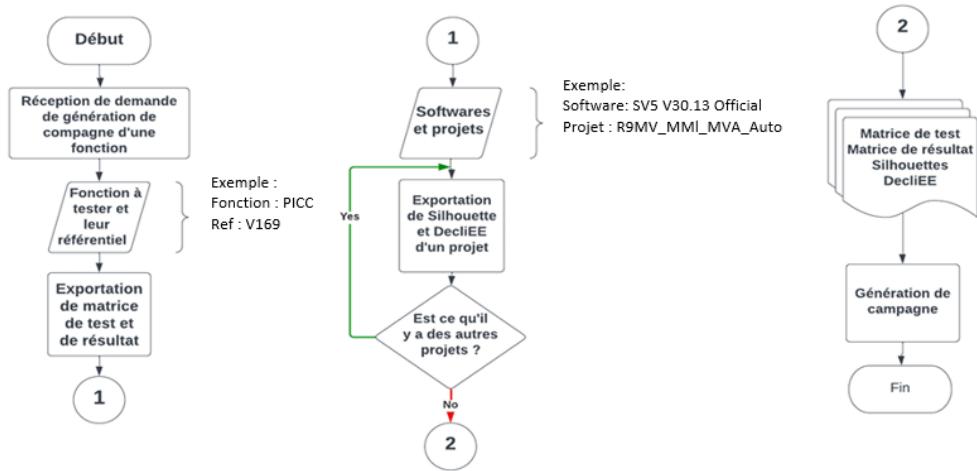


FIGURE 1.7 – Diagramme de processus de génération de campagne

Voici un extrait de document de campagne :

1.4. CADRAGE DE PROJET

Function(s) or type of test	1	2	3	4	Thématiques globales :	5	6	7	8	9	10	11	12	
Project :	SILHOUETTE DECLIEE				Thématiques initiales A					Thématiques initiales NA				
Software version :	V1.0				Non silhouettes initiales					Thématiques NA				
Test Book(s) version :	V1.0				Télécodages communs					Télécodages communs				
Type de bench :	V1.0				Créer fichier télécodage									
Thématique(s) de la campagne :	OK													
Résultat :	NOK													
	NT													
	OK													
N° Fiches de test	1				Descriptif du test					Indice test				
Catégorie	Télécodages spécifiques				Thématique(s)									
	Télécodages spécifiques													
	Thématiques initiales A													
	Thématiques initiales NA													
	Non silhouettes initiales													
	Thématiques NA													
	Télécodages communs													
	Créer fichier télécodage													
	Thématiques incompatibles													
	Thématiques applicables													
	Thématiques non applicables													
	Télécodages non applicables													
	Télécodages communs													
	Créer fichier télécodage													
	Télécodages spécifiques													
	Thématiques initiales A													
	Thématiques initiales NA													
	Non silhouettes initiales													
	Thématiques NA													
	Télécodages communs													
	Créer fichier télécodage													
	Thématiques incompatibles													
	Thématiques applicables													
	Thématiques non applicables													
	Télécodages non applicables													
	Télécodages communs													
	Créer fichier télécodage													
	Télécodages spécifiques													
	Thématiques initiales A													
	Thématiques initiales NA													
	Non silhouettes initiales													
	Thématiques NA													
	Télécodages communs													
	Créer fichier télécodage													
	Thématiques incompatibles													
	Thématiques applicables													
	Thématiques non applicables													
	Télécodages non applicables													
	Télécodages communs													
	Créer fichier télécodage													
	Télécodages spécifiques													
	Thématiques initiales A													
	Thématiques initiales NA													
	Non silhouettes initiales													
	Thématiques NA													
	Télécodages communs													
	Créer fichier télécodage													
	Télécodages spécifiques													
	Thématiques initiales A													
	Thématiques initiales NA													
	Non silhouettes initiales													
	Thématiques NA													
	Télécodages communs													
	Créer fichier télécodage													
	Télécodages spécifiques													
	Thématiques incompatibles													
	Thématiques applicables													
	Thématiques non applicables													
	Télécodages non applicables													
	Télécodages communs													
	Créer fichier télécodage													
	Télécodages spécifiques													
	Thématiques initiales A													
	Thématiques initiales NA													
	Non silhouettes initiales													
	Thématiques NA													
	Télécodages communs													
	Créer fichier télécodage													
	Télécodages spécifiques													

pagne et les fichiers contenant les informations sur les thématiques et leur applicabilité dans les projets entraîne une incohérence. Toutes ces informations doivent être mentionnées dans livrable qui sera envoyé pour la mise à jour de silhouette et éliminer les incohérences déclarées pour que les responsables de la phase de validation puissent commencer les tests sur le software.

Après validation de la campagne, l'équipe de validation HIL se concentre sur la configuration du banc sur le logiciel. Ce logiciel simule toutes les fonctionnalités du véhicule. Lors de la configuration, l'intégration de la silhouette est essentielle pour présenter les options activées et désactivées (télécodage) au niveau du véhicule en cours de développement. Une fois cette configuration terminée, l'étape de validation commence [2].

1.4.1.4 Validation

Après la mise à jour des fichiers, la phase de validation se déclenche en appliquant les scénarios conçus dans les plans de test en se basant sur les exigences du client. Cela permet de vérifier la conformité de fonctionnement des sorties obtenues avec les données spécifiées dans les documents d'entrée. Cette opération est réalisée à travers un banc connecté à distance.

Mais avant d'entamer l'implémentation des scénarios de test, il faut tout d'abord s'assurer de l'applicabilité des thématiques (les options) liées aux exigences client dans le fichier DECLIEE du projet (ce qu'on va traiter par la suite dans ce projet), par ce que tester une fonctionnalité non applicable reviendrait à évaluer un comportement inexistant, ce qui fausse les résultats et alourdit inutilement les campagnes de test. Voici ci-dessous deux exemples des exigences client :

1.4. CADRAGE DE PROJET

N° Requirement		Content of the Requirement
REQ-0111111 A State : Release Modified : 11/16/2023 1:47:25 PM AEN-VHL-AC-3388 Owner : M501516 Generic : Yes Maturity : Robust Grade : Key Requirement : Yes Flexibility : Abstraction Level : AAA ISAA Compliance : Yes AAA SSTG : Comments		The GERER_P function shall set the DISPO_ESTAT_RECH flow to DISPONIBLE as soon as the function has at least respectively computed one of each following flow. ETAT__STAB_RECH_H_DEBUT ETAT__STAB_TYPE_RECH
Effectivity Expression		HP-0000873[A-∞] AND (HP-0000873[AFC(AFC_02)] AND HP-0000873[AYQ(AYQ_01)]) OR (HP-0000873[AYQ(AYQ_01)] AND HP-0000873[AFC(AFC_03)]) OR (HP-0000873[AYQ(AYQ_02)] AND HP-0000873[AXD(AXD_03)]) OR (HP-0000873[AXD(AXD_04)] AND HP-0000873[AYQ(AYQ_02)]) OR (HP-0000873[AYQ(AYQ_02)] AND HP-0000873[DXD(AXD_05)])
LCDV Simplified EN		HP-0000873 [DICO MULTIGAMME T2_2016 - ∞] AND AYQ TYPE_DIV(AYQ_01 BEFORE_FUNCT_C) AND AFC TYPE_CHAINE_R(AFC_02 HY,AFC_03 ELEC) OR AYQ TYPE_DIV(AYQ_02 FUNCT_D) AND AXD TYPE_OF R(AXD_03 PHEV,AXD_04 ELECTRIC,AXD_05 MHEV)
Input Requirement		AEN-AHLST-72(2.0)

FIGURE 1.10 – Première forme des exigences d'une fonction X

REQ-0222222 C			AEN-DHL-AC-INFO-8583(1)	Modified : 01/07/2025
State : Release	Owner : M501518		Grade :	
Generic : Yes	Flexibility :		Key Requirement : Yes	
AAA AATG :	Maturity : Robust		Cyber Relevant : No	
Content of the Requirement				
Management of the memorized speeds setting in case of the ALV function presence:				
IF				
The user informs a AVV instruction speed memorized selectable through a tactile telematic box (VAT_AVV_N) (n from 10 to 15) lower than VITESSE_CA				
THEN				
The speed is not <u>taken into account</u> and the display speed is equal to VITESSE_CA				
Description				
LI-EVOL-0033 : Evolution of the diversity SPEED				
Diversity Expression				
HP-0000873 [DICO MULTIGAMME T2_2016 - ∞] AND AYQ TYPE_DIV(AYQ_02 FUNCT_C) AND C16 SPEED (C16_02 ICA PROPOSED SPEEDS,C16_03 ICA PROPOSED SPEED+ANTICIPATED,C16_04 ICA AUTOMATIC) OR AZA OPTION_ALV(AZA_01 WITH) AND AYQ TYPE_DIV(AYQ_01 BEFORE_FUNCT_C)				

FIGURE 1.11 – Deuxième forme des exigences d'une fonction Y

Après être sûr de l'applicabilité des thématiques (voir si-dessous) dans le projet, les scénarios de test, présenté dans le plan de test développé par l'équipe de plan de test (en Inde), sont appliqués un par un. Chaque scénario est testé sur le logiciel, en se concentrant sur le comportement externe du système, sans prendre en compte les détails internes. Cette méthode repose sur le principe que le testeur n'a pas besoin de

1.5. PRÉSENTATION DU PROJET

connaître les détails internes du système pour évaluer son bon fonctionnement.

Le fichier de plan de test, qui contient toutes les informations nécessaires pour l'implémentation des tests, est utilisé comme référence. Les résultats obtenus durant la validation doivent être compatibles avec les sorties et les résultats attendus spécifiés dans le fichier de test. Voici un extrait ci-dessous du plan de test :

1	2	3	4	5	6	7	8	9	10	11	12
1	IN DU TEST	B9107_GC_01_01_0001	TESTCASE	5							
2	TITRE DU TEST										
3	BUT DU TEST	Verify if the fuel is too old (Soundtype B41+ ASIL A)									
4	REQUISITE										
5	FEPS	REQ-0615839(A) REQ-0115844(A) REQ-0115857(A) REQ-0715859(A) REQ-0415868(A) REQ-0715884(A)									
6	CATEGORIE	AUTO	PONDÉRATION	P1							
7	STATUT	VALUÉE	MOT CLE								
8	THEMATIQUE 2	ATD_01 AZT_01 AWF_00 LUH_01 CLL_01									
9	THEMATIQUE 1	ATD_01 AWF_00 LUH_01 CLL_01									
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											
22											
23											
24											
25											
26											
27											
28											
29											
30											

FIGURE 1.12 – Exemple de fichier de plan de test

Dans ce fichier, on trouve pour chaque fonction les paramètres d'entrée avec leurs valeurs ainsi que les sorties attendues. En cas d'incohérence détectée lors du test, il est nécessaire de vérifier les états des thématiques dans les fichiers silhouette, car le problème pourrait être lié à cela. De plus, il est important de vérifier si le problème provient d'une évolution de version non prise en compte par l'équipe de plan de test au niveau des exigences.

1.5 Présentation du projet

Dans le contexte de la conception et du développement des architectures électriques embarquées, le BSI joue un rôle central. Il regroupe de nombreuses fonctions électriques et électroniques et varie selon les projets véhicules. Chaque projet impose un ensemble d'options fonctionnelles spécifiques, définies par des exigences clients. Le suivi de ces options, leur analyse et leur adaptation aux différents projets nécessitent une

rigueur importante, particulièrement lors de la phase de validation et de configuration des fonctions embarquées.

1.6 Contexte du projet

Dans le processus de validation des systèmes embarqués, et plus particulièrement dans le cadre des campagnes de validation du BSI, une étape cruciale précède l'exécution des plans de test : il s'agit de vérifier que les thématiques (options) associées aux exigences à tester sont bien applicables dans le projet concerné.

En effet, certaines anomalies "NOK" détectées en validation peuvent provenir non pas d'un défaut réel, mais d'un test réalisé sur une fonction non applicable dans le contexte du projet. Cela peut fausser les résultats, générer des analyses inutiles, voire entraîner la création de défauts non justifiés.

D'où la nécessité, avant toute analyse de NOK ou création de défaut, d'identifier les thématiques liées au cas du test, et de s'assurer que ces thématiques sont bien définies comme "applicables" dans le projet ciblé.

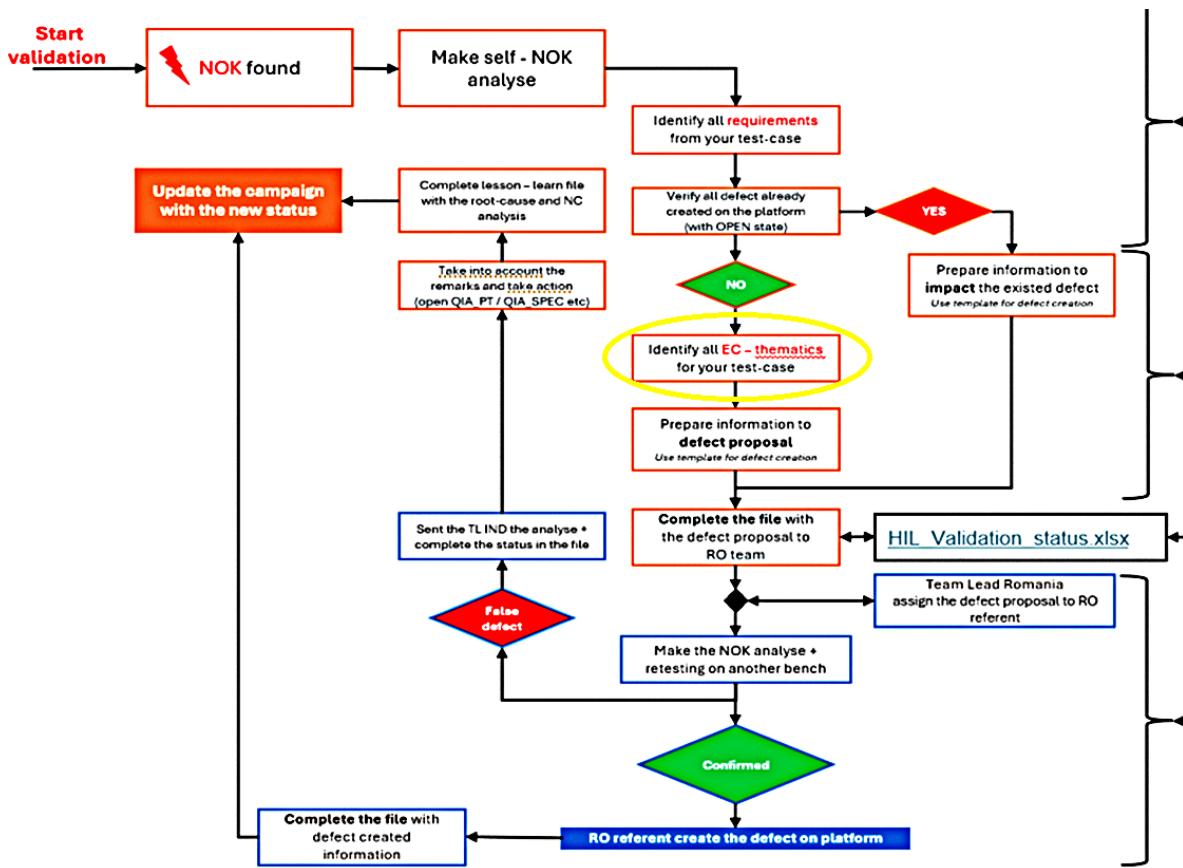


FIGURE 1.13 – Processus de validation de BSI

1.7 Problématique

Dans le cadre du développement des logiciels embarqués pour le BSI, les exigences client définissent les options fonctionnelles attendues sur un véhicule donné. Ces options sont généralement exprimées sous forme de thématiques (options), regroupées selon des expressions logiques complexes. Très souvent, plusieurs exigences doivent être prises en compte simultanément, ce qui engendre un grand nombre de combinaisons possibles de thématiques à vérifier.

Le problème principal réside dans la vérification manuelle de l'applicabilité de toutes ces thématiques par rapport à un ou plusieurs projets. Cette opération est :

- Chronophage, car elle nécessite de croiser une grande quantité d'informations issues de documents différents ;
- Complexe, en raison de la logique entre les thématiques ;
- Sujette aux erreurs, notamment lorsque les combinaisons sont longues quand plu-

sieurs exigences se combinent.

Pour bien encadrer et résumer la problématique et la démarche à suivre nous avons utilisé la méthode dite QQOQCP qui permet d'analyser toutes les dimensions du problème :

Quoi ?	Amélioration et optimisation des processus de validation du calculateur BSI
Qui ?	Le projet concerne l'équipe de validation HIL de BSI
Où ?	Expleo Group Tanger, département ESE
Quand ?	6 mois de période de stage
Comment ?	Méthode Agile
Pourquoi ?	Pour automatiser une tâche qui se fait actuellement manuellement afin de réduire le risque d'erreurs et de gagner du temps et augmenter l'efficacité.

TABLE 1.2 – Méthode QQOQCP

1.8 Objectifs

Parmi les objectifs principaux de ce projet :

- Accélérer le traitement des exigences, en réduisant le temps consacré à l'analyse manuelle ;
- Etablir une vérification automatique d'applicabilité des thématiques (options) dans un ou plusieurs projets ;
- Réduire les risques d'erreurs humaines dans la validation fonctionnelle ;
- Augmenter l'efficacité de processus de validation.

1.9 Solution proposée

La solution proposée à la problématique citée est de concevoir et développer une application logicielle permettant de traiter automatiquement les exigences fonctionnelles liées au calculateur BSI, et de vérifier l'applicabilité des thématiques dans un ou plusieurs projets.

1.10 Besoins fonctionnels

Gestion des exigences : L'utilisateur doit avoir la possibilité d'ajouter de nouvelles exigences ou de supprimer des exigences existantes. L'application doit offrir une interface permettant de visualiser l'ensemble des exigences et de chercher facilement une exigence précise et aussi assurer un suivi des nouvelles versions des exigences. Ainsi qu'il faut traiter différents formes des exigences à partir de différents documents, contrôler le cas des exigences vides (génériques), et aussi résoudre le problème de la redondance des exigences dans plusieurs documents .

Comparaison des exigences : L'application doit être capable de comparer automatiquement les différentes exigences afin d'avoir toutes les combinaisons possibles et d'éliminer toutes redondances éventuelles.

Comparaison avec les projets : L'application doit permettre de vérifier l'applicabilité d'une ou d'un ensemble d'exigences à un ou à plusieurs projets simultanément.

Gestion des projets : L'application doit permettre l'ajout de nouveaux projets, ainsi que la suppression de projets existants en cas de la publication de nouvelles versions.

1.11 Besoins non fonctionnels

Performance et réactivité : L'application doit garantir une bonne réactivité, c'est-à-dire que chaque action de l'utilisateur, comme l'importation de grands documents (qui est une sous-problématique qu'on a traité par la création d'une base de données), l'extraction de données, la recherche ou la comparaison, doit être traitée rapidement.

Fiabilité et robustesse : L'application doit être conçue pour gérer proprement toutes les erreurs possibles par l'utilisateur.

Traçabilité : Toutes les actions importantes par l'utilisateur doivent être enregistrées.

Sécurité : Les accès à l'application doivent être assurés par un système d'authentification fiable, et aussi il faut sécuriser toutes actions critiques possibles par l'utilisateur.

Maintenabilité et évolutivité : Le code de l'application doit être structuré par des modules et commenté de façon à pouvoir être facilement modifié ou enrichi par la suite.

Ergonomie : L'interface doit être claire, intuitive et agréable à utiliser, et permet un accompagnement à l'utilisateur dans ses actions.

Portabilité : L'application doit fonctionner sur toutes machines à l'aide d'un fichier exécutable.

1.12 Contraintes techniques

- Langage principal : Python 3
- Framework UI : PyQt5
- Manipulation de fichiers Word : python-docx
- Manipulation de fichiers Excel : pandas
- Base de données : SQLite

1.13 Déroulement du projet

Dans cette partie, on va planifier les étapes de notre projet, chaque étape avec sa date de départ, sa date finale et sa durée, en utilisant le tableau des tâches.

1.13.1 Tableau des tâches

Tâche	Date de départ	Date de fin	Durée (jours)
Cahier de charge	03/03/2025	06/06/2025	70
Conception	11/03/2025	02/06/2025	60
Développement	20/03/2025	11/06/2025	60
Testing	20/03/2025	11/06/2025	60
Formation avec l'équipe validation de BSI	07/04/2025	06/06/2025	45
Rédaction du rapport	07/05/2025	16/06/2025	29
Présentation	17/06/2025	19/06/2025	3

TABLE 1.3 – Tableau des tâches

1.13.2 Clarification des tâches

Cahier des Charges : Le cahier des charges est une étape cruciale dans la planification du projet. Il encadre le projet en définissant clairement la problématique, les objectifs à atteindre, les besoins fonctionnels, les besoins non fonctionnels, et les contraintes à respecter. Cette documentation inclut une évaluation des exigences techniques et fonctionnelles nécessaires pour concevoir l'application, garantissant ainsi que le projet est bien structuré.

Conception de l'application : Au cours de la phase de conception, nous avons développé l'architecture de l'application en UML (Unified Modeling Language). De plus,

nous avons identifié les outils requis (language de programmation, environnement de développement, frameworks...), tout en assurant que l'équipe projet avait la formation et les compétences nécessaires pour les utiliser efficacement.

Développement : Cette phase consiste en l'écriture du code ainsi que sa structuration et son organisation en modules, afin de permettre toute mise à jour éventuelle.

Testing : Des tests rigoureux ont été menés pour évaluer les performances tout au long de l'étape de développement.

Formations avec l'équipe de validation BSI : Des formations avec l'équipe de validation BSI étaient nécessaires pour comprendre l'impact de projet au sein de l'équipe.

Rédaction du rapport : L'avant dernière phase du projet consiste à documenter toutes les étapes de conception, développement, les résultats des tests, et les optimisations réalisées. La rédaction était au fur à mesure avec le développement du projet.

Présentation : Pour résumer le rapport, nous avons effectué une présentation Power Point afin de mettre en évidence les points primordiaux de notre projet.

1.13.3 Méthodologie

En termes de méthodologie du projet, nous avons opté pour l'approche Agile Scrum. Cette méthodologie a été déterminante dans la réussite de notre projet. Cela nous a permis de maintenir un flux de travail dynamique et adaptatif tout au long du cycle de vie du projet. Dans le cadre de l'Agile Scrum, nous organisons des réunions régulières pour que tout le monde soit au courant. Nous tenons des réunions hebdomadaires pour discuter de l'ensemble des progrès, relever les défis et planifier les tâches de la semaine à venir. Cette approche collaborative a contribué de façon significative à l'efficacité et à la capacité d'adaptation de notre projet.

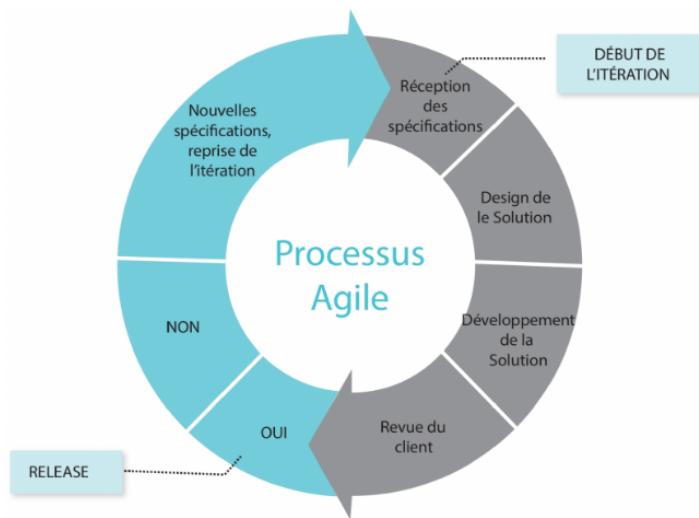


FIGURE 1.14 – Processus Agile

1.14 Conclusion

Ce chapitre a permis de poser le cadre général du projet en présentant l'entreprise d'accueil Expleo Group, son positionnement dans l'industrie automobile, et plus particulièrement son expertise dans le domaine des systèmes embarqués. Nous avons également détaillé le contexte spécifique du projet mené durant le stage, en mettant en évidence les enjeux liés à la validation du calculateur BSI, ainsi que les limites des méthodes actuelles. Enfin, les objectifs, les besoins, les contraintes techniques, et la planification ont été clarifiés, posant ainsi des bases solides pour le développement d'une solution logicielle adaptée et efficace.

CHAPITRE 2

ÉTAT DE L'ART DES SYSTÈMES AUTOMOBILES

2.1 Introduction

Ce chapitre expose l'état de l'art des systèmes automobiles, en offrant une perspective sur les différentes notions abordées au cours du projet, telles que les types de véhicules, l'architecture électronique et électrique, le BSI, ainsi que les tests sur bancs HIL. L'ensemble de ces éléments est présenté dans le cadre du processus de validation du calculateur BSI, dans lequel s'inscrit ce projet.

2.2 Les types de Véhicules

Avec la transition vers une mobilité plus propre et plus durable, l'industrie automobile propose aujourd'hui plusieurs technologies de motorisation alternatives aux moteurs thermiques classiques. Ces véhicules électrifiés se distinguent par le degré d'hybridation et le mode d'alimentation en énergie. On distingue :

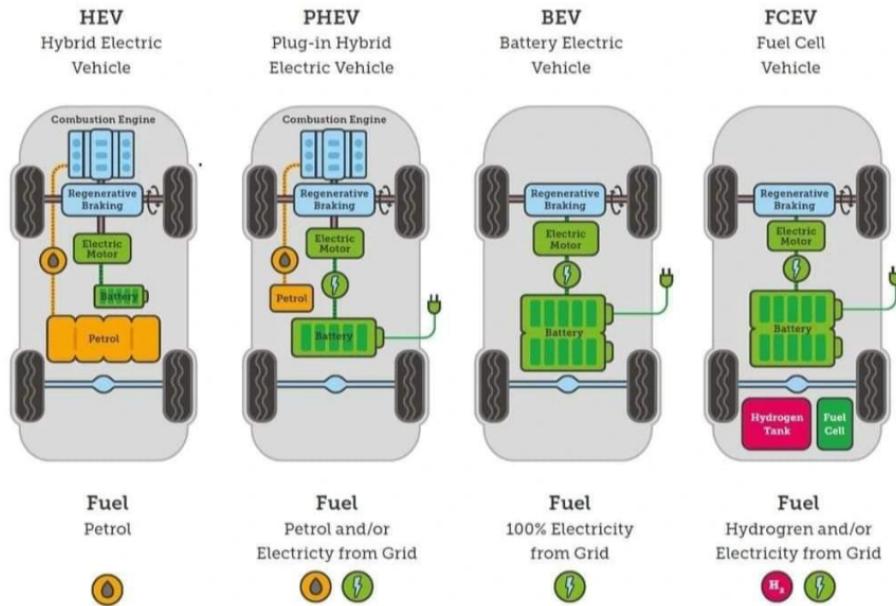


FIGURE 2.1 – Différents types d'architecture de véhicule électrique

2.2.1 HEV

Les HEV (Hybrid Electric Vehicle) combinent un moteur à combustion interne (essence ou diesel) et un ou plusieurs moteurs électriques. L'énergie électrique est stockée dans une batterie qui se recharge automatiquement grâce au moteur thermique et au freinage régénératif. Les HEV ne nécessitent pas de recharge externe. Ce système permet de réduire la consommation de carburant et les émissions polluantes, tout en offrant une autonomie similaire à celle des véhicules conventionnels [3].

2.2.2 MHEV

Les MHEV (Mild Hybrid Electric Vehicle) utilisent un petit moteur électrique pour assister le moteur thermique, mais ne peuvent pas rouler uniquement à l'électricité. Ce moteur électrique est alimenté par une batterie 48V (parfois 12V) qui se recharge grâce à la récupération d'énergie au freinage. Il améliore les phases de démarrage, de redémarrage (start-stop), et peut fournir un surcroît de couple à basse vitesse, réduisant légèrement la consommation de carburant et les émissions. Le MHEV est une solution intermédiaire peu coûteuse qui permet aux constructeurs de respecter les normes d'émissions sans perturber la conception des véhicules[3].

2.2.3 PHEV

Les PHEV (Plug-in Hybrid Electric Vehicle) fonctionnent de manière similaire aux HEV, mais disposent d'une batterie plus grande qui peut être rechargée via une prise électrique externe. Cela leur permet de parcourir des distances plus longues en mode tout électrique, généralement entre 30 et 60 km, avant que le moteur thermique ne prenne le relais. Les PHEV offrent ainsi une flexibilité accrue, permettant de réduire significativement la consommation de carburant et les émissions lors des trajets quotidiens, tout en conservant la possibilité de longs trajets sans souci d'autonomie[3].

2.2.4 BEV

Les BEV (Battery Electric Vehicle) sont entièrement propulsés par un ou plusieurs moteurs électriques alimentés par une batterie rechargeable. Ils ne possèdent pas de moteur à combustion interne, ce qui signifie qu'ils n'émettent aucune émission polluante lors de leur utilisation. Les BEV nécessitent une recharge régulière via des bornes de recharge domestiques ou publiques. Grâce aux avancées technologiques, leur autonomie peut atteindre jusqu'à 500 km, rendant ces véhicules adaptés à une utilisation quotidienne et à des trajets plus longs, tout en contribuant à la réduction de l'empreinte carbone.

2.2.5 FCEV

Les FCEV (Fuel Cell Electric Vehicle) utilisent l'hydrogène comme source d'énergie pour produire de l'électricité via une réaction chimique entre l'hydrogène et l'oxygène, ne rejetant que de la vapeur d'eau comme sous-produit. Cette électricité alimente un moteur électrique qui propulse le véhicule. Les FCEV offrent une autonomie comparable à celle des véhicules thermiques et un temps de ravitaillement rapide. Cependant, leur adoption est actuellement limitée en raison de l'infrastructure de ravitaillement en hydrogène encore peu développée et du coût élevé de la technologie[4].

2.3 Architecture Electrique Electronique

L'évocation des calculateurs et des architectures automobiles ne saurait se faire sans mentionner deux concepts clés : les systèmes embarqués et le temps réel.

On désigne comme un « système embarqué » un système électronique et informatique indépendant, conçu pour accomplir une mission spécifique, souvent en temps réel, qui se caractérise par sa compacité et sa faible consommation d'énergie. À moins qu'il ne

soit requis par le système auquel il appartient, l'appareil embarqué ne dispose ni de clavier, ni d'écran, ni de connexions série, ni de système de stockage, ni d'interface utilisateur. On évoque un système en temps réel quand ce dernier est en mesure de gérer et d'opérer un processus physique à une cadence qui correspond à la progression du processus surveillé.

2.3.1 Les calculateurs automobiles

Dans le domaine de l'automobile, le terme ECU fait référence à un système embarqué ou calculateur embarqué qui contrôle des dispositifs physiques à l'intérieur d'un véhicule. L'ECU renvoie des directives pour différents systèmes électriques, indiquant comment agir et comment opérer.

L'image ci-dessous présente certains calculateurs des automobiles Stellantis.

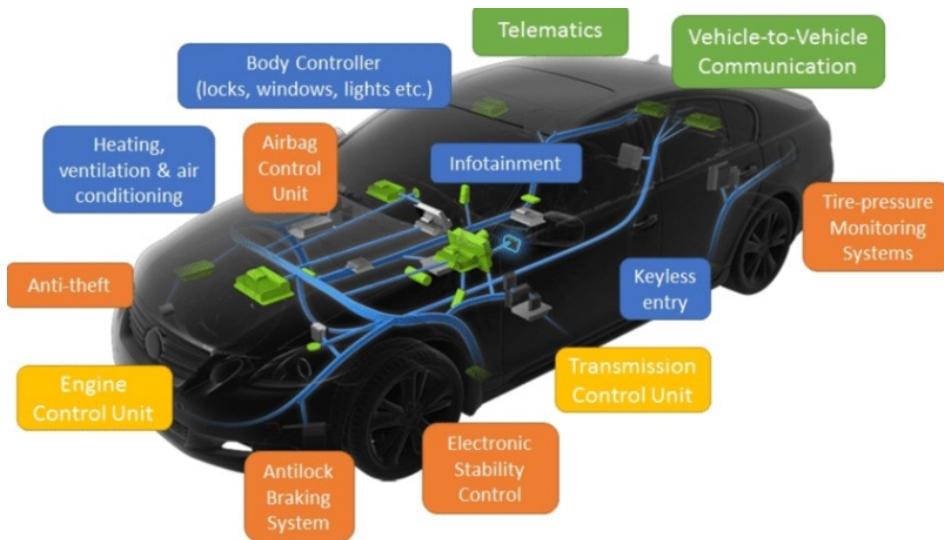


FIGURE 2.2 – Quelques calculateurs Automobiles

On distingue plusieurs sortes de calculateurs, tels que le module de contrôle du moteur, également connu sous l'acronyme ECM (Engine Control Module), le module de contrôle du groupe motopropulseur appelé PCM (Powertrain Control Module) et le module de commande des freins, désigné par BCM (Brake Control Module).

Les véhicules récents peuvent comporter jusqu'à 80 ECU, et du fait de leur complexité grandissante, la programmation pour leur développement devient plus difficile à gérer.

2.3.2 Bus de communication

Dans un véhicule, divers bus de communication sont employés pour faciliter le partage d'informations entre les différents éléments électroniques et les calculateurs intégrés.

Ces bus de communication ont une importance cruciale dans l'opération du véhicule en facilitant le transfert de données et d'instructions entre les systèmes électroniques comme le moteur, la transmission, les systèmes de sécurité, le système de divertissement, ainsi que les capteurs et les actionneurs. Dans les véhicules modernes, on peut souvent trouver les bus de communication suivants :

- **Bus LIN (débit maximal de 20 kbps)** : Réservé aux systèmes non essentiels comme la fermeture des portes, les miroirs rétroviseurs ou le toit panoramique. C'est un bus très lent, mais économique.

- **Bus CAN (Débit maximal de 1Mbps)** : Affecté aux systèmes de confort, de gestion du moteur et de la transmission, etc.

Bus Flexray (Débit maximal de 10 Mbps) : Destiné au système de freinage ou à la sécurité avancée. Il se démarque du CAN par sa vitesse de transmission et sa fiabilité, mais aussi par un coût supérieur.

- **Bus MOST (25 ou 50 Mbps)** : réservé aux systèmes de communication entre le système mains libres et l'autoradio ou le lecteur DVD par exemple. Il s'appuie sur la technologie de la fibre optique, ce qui le rend onéreux et complexe à déployer [5].

2.3.3 Protocole LIN

Le protocole de communication de données LIN est principalement employé dans le secteur automobile pour relier divers capteurs, actionneurs et calculateurs au sein d'un véhicule.

Le protocole LIN a été élaboré pour être à la fois simple et peu coûteux, utilisant une unique paire de fils pour le transfert de données. Ce dispositif autorise les appareils à communiquer des données à un faible débit, souvent de l'ordre du kilobit par seconde, tout en consommant peu d'énergie et en offrant une grande fiabilité.

Le protocole LIN est généralement employé pour des applications de contrôle élémentaire, comme l'éclairage, les systèmes de ventilation, les capteurs thermiques, les capteurs de pression et d'autres équipements comparables. Il est aussi employé dans d'autres secteurs industriels où une transmission de données simple et économique est indispensable.

2.3.3.1 Format des trames LIN

Le bus LIN est un type de bus utilisé pour la communication entre un maître unique et un ou plusieurs périphériques qui agissent en tant qu'esclaves. C'est le maître qui gère intégralement la communication avec le bus LIN. La trame, qui est segmentée en un en-tête et en réponse, constitue l'élément fondamental de communication sur le bus LIN. Le nœud maître envoie toujours l'en-tête, qui comporte trois champs distincts : le break, la sync (synchronisation) et l'ID (identifiant).

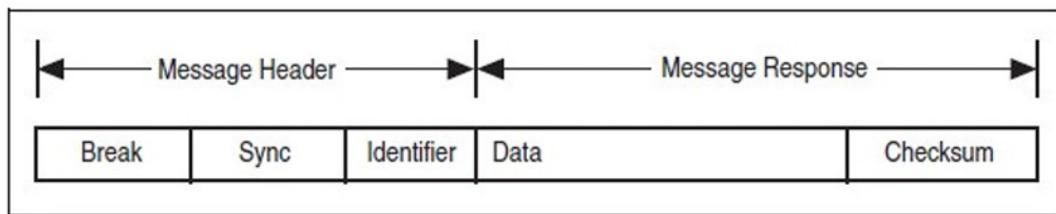


FIGURE 2.3 – Formats des trames LIN

- **Break** : Chaque trame LIN commence par un break qui inclut 13 bits (nominal) de poids fort, suivi d'un délimiteur break sur un bit (nominal) de poids faible. Il fonctionne comme un message d'initialisation pour l'ensemble des nœuds du bus.

- **Sync** : Le champ de synchronisation est le second champ transmis par la tâche principale dans l'en-tête. On définit Sync comme le caractère x55. Le champ de synchronisation donne la possibilité aux équipements esclaves de réaliser une détection automatique du taux de transfert pour adapter leurs vitesses en vue d'une synchronisation avec le bus.

Identifier : Le champ ID est le dernier élément envoyé par la tâche maître dans l'en-tête. Il sert à identifier chaque message circulant sur le réseau et définit quels nœuds doivent le recevoir ou y répondre. Les tâches esclaves surveillent en permanence les champs ID, vérifient leur parité et déterminent si elles doivent agir en tant qu'émetteur ou récepteur pour un identifiant donné.

Le bus LIN prend en charge 64 ID répartis comme suit :

ID 0 à 59 : utilisés pour les trames de données standard.

ID 60 et 61 : dédiés aux données de diagnostic.

ID 62 : réservé aux extensions personnalisées.

ID 63 : conservé pour les futures évolutions du protocole.

L'ID est transmis sous la forme d'un octet protégé : les 6 premiers bits représentent l'identifiant brut, tandis que les 2 derniers bits contiennent la parité.

Data : Le champ de bytes d'information est envoyé par la tâche esclave dans la réponse.

Ce champ peut contenir entre 1 et 8 octets d'informations.

Checksum : Le champ de vérification est envoyé par la tâche esclave dans la réponse. Le bus LIN détermine l'emploi de l'un des deux algorithmes de somme de contrôle afin de déterminer la valeur du champ somme de vérification sur 8 bits. La somme de contrôle standard se fait par l'addition unique des octets de données, tandis que la somme de contrôle améliorée est obtenue en combinant les octets de données avec l'ID protégé.

2.3.3.2 Avantages et Limitations de LIN

Le protocole LIN présente plusieurs avantages et limitations, parmi :

Les Avantages du LIN :

- Protocole Standardisé.
- Gain économique par effet de volume.
- Ségrégation des fonctions, aspect modulaire.
- Réseau déterministe : Garantie des temps de latence.

Les limitations du LIN :

- Pas d'acquittement.
- Pas de mécanisme de réémission.
- Débit limité.
- Réseau déterministe : pas de flexibilité d'émission des messages [5].

2.3.4 Le LIN dans les architectures Stellantis

Le bus LIN est utilisé en parallèle avec le bus CAN, qui est principalement employé pour communiquer avec des capteurs ou des micro-actionneurs à faible vitesse, tels que les vitres électriques, le toit ouvrant ou la mesure de l'intensité de la luminosité ambiante pour l'allumage automatique des phares.

LIN est une version simplifiée et abordable du bus CAN, proposant une plus grande facilité d'utilisation et un meilleur rendement en termes de ressources.

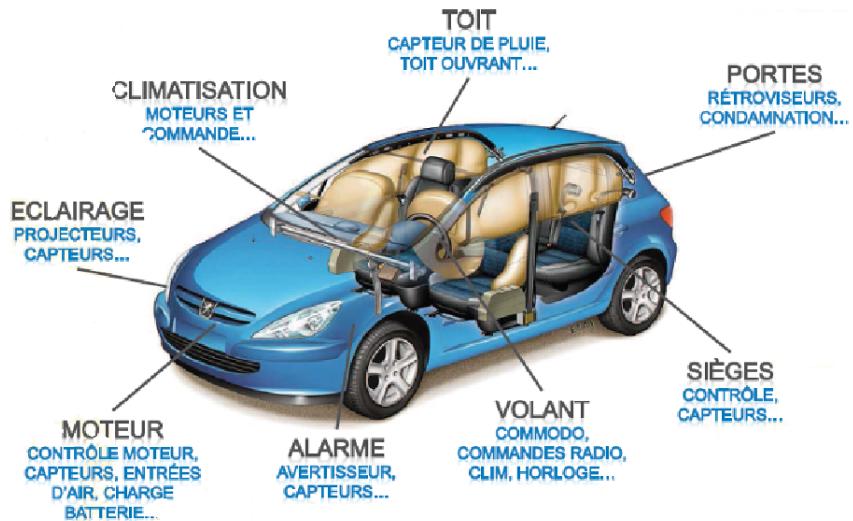


FIGURE 2.4 – Le LIN dans les architectures Stellantis

2.3.5 Protocole CAN

Le CAN a vu le jour en 1990, en réponse aux exigences de l'industrie automobile face à l'expansion de l'électronique embarqué. En 2005, une voiture moyenne est équipée d'environ une centaine de microcontrôleurs. Afin d'éviter l'utilisation de 2 kilomètres de câblage pour une voiture moderne, équivalente à 100 kg de cuivre, nous devions concevoir un bus série qui facilite grandement l'incorporation des fils dans le châssis.

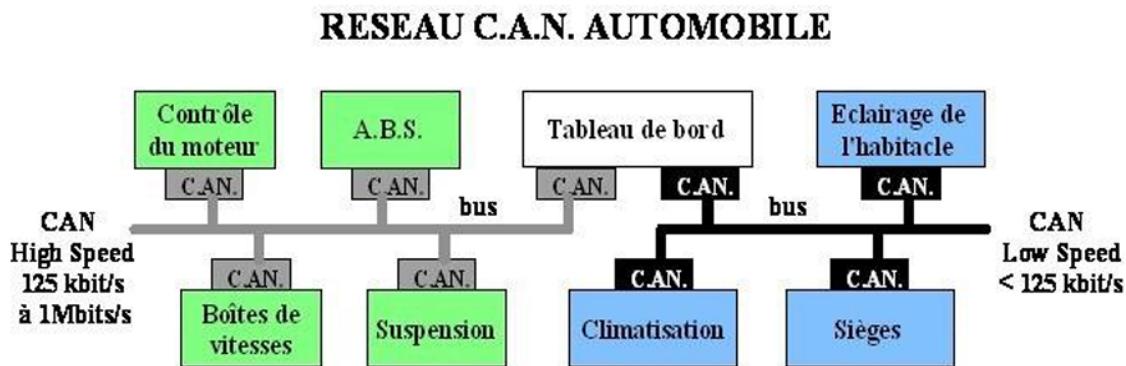


FIGURE 2.5 – Réseau CAN Automobile

En pratique, les voitures utilisent généralement deux bus CAN principaux, avec des débits différents :

CAN HS (High Speed - 500 kbit/s à 1 Mbit/s) pour les systèmes critiques :
Sécurité : ABS, airbags, freinage d'urgence.

CAN LS (Low Speed - jusqu'à 125 kbit/s) pour les fonctions non urgentes :
Accessoires : éclairage, climatisation, sièges électriques.

Il existe sous deux versions :

CAN2.0A : trame standard identificateur de 11 bits (CAN standard).

CAN2.0B : trame plus longue avec identificateur sur 29 bits (CAN étendu).

2.3.5.1 Les signaux du bus CAN

La transmission des données se fait via une paire de fils différentielles. Il y a donc deux fils qui constituent la ligne. Les niveaux logiques (dominants et récessifs) sont déterminés par la différence de potentiel entre les deux lignes CAN L et CAN H. Les niveaux de tension sur CANL et CANH sont déterminés par la version Low Speed ou High Speed du bus. Ces niveaux de tension sont associés à un codage connu sous le nom de NRZ (No Return to Zero : il n'y a jamais d'absence de courant sur la ligne). L'usage de la masse n'est plus nécessaire et les niveaux logiques représentent désormais deux niveaux de tensions différents.

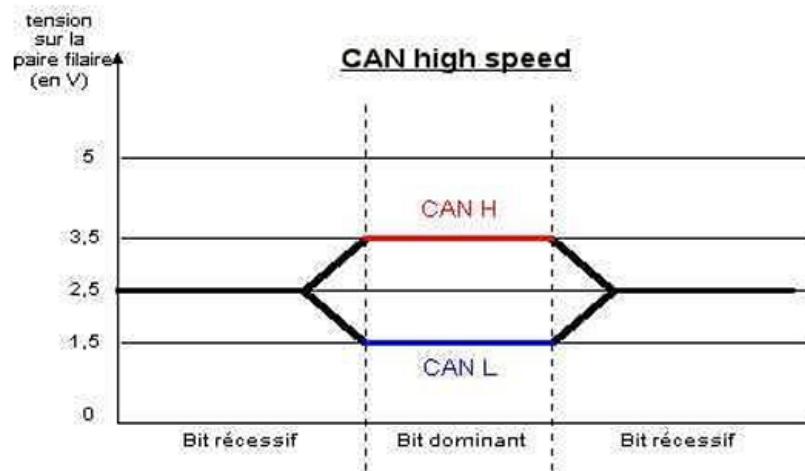


FIGURE 2.6 – CAN High Speed

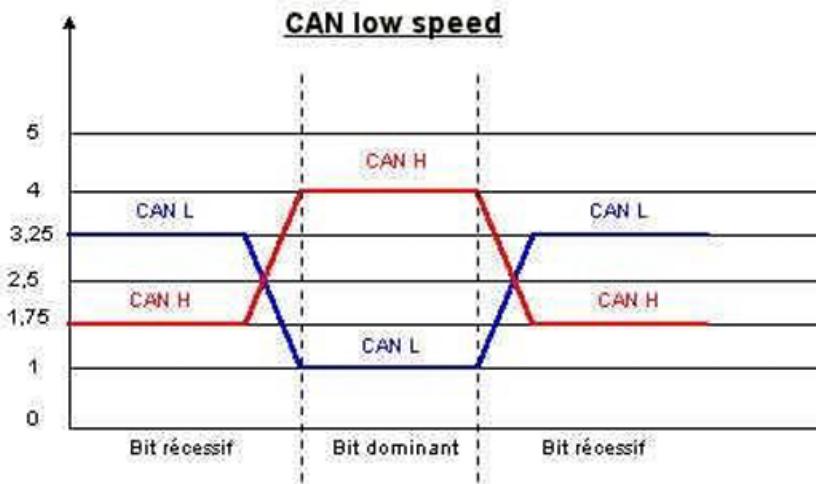


FIGURE 2.7 – CAN Low Speed

2.3.5.2 La longueur des bus CAN

La longueur maximale du bus est spécifiée par la charge capacitive et le taux de transmission. Voici les configurations suggérées, en tenant compte que la norme impose une vitesse maximale de 1Mbit/s :

Débit	Longueur
1Mbit/s	40m
500Kbit/s	100m
100Kbit/s	500m
20Kbit/s	1 000m

FIGURE 2.8 – La relation Débit/Longueur dans les bus CAN

2.3.5.3 Le Diagnostic

Le diagnostic est une caractéristique logicielle que l'on incorpore dans un calculateur afin d'assurer les opérations suivantes :

- La configuration des fonctions (ajustement des paramètres de fonctionnement, télécodage).
- Les ajustements des calculateurs pour le véhicule (Calibrations).

- La vérification du fonctionnement des calculateurs et des composants associés (Informations provenant d'un capteur, informations internes au calculateur, test d'actionneur).
- Téléchargement des logiciels (versions récentes).
- La réalisation de certaines procédures de maintenance(apprentissage de clé, etc.)
- La lecture des défaillances que le calculateur peut détecter (capteur défaillant, données incohérentes, information hors des plages de validité...).

Les protocoles de diagnostic les plus couramment employés sont le KWP2000 (Keyword Protocol 2000) selon la norme ISO 14230 et l'UDS (Unified Diagnostic Services) selon la norme ISO 14229, ce dernier étant aujourd'hui le plus répandu. Le diagnostic s'effectue sur le réseau CAN, et en général, un ID CAN distinct est alloué à chaque ECU pour des objectifs de diagnostic.

Le protocole de diagnostic repose sur les services de diagnostic, chaque service ayant une fonction précise. Un service diagnostic est réalisée par l'échange d'une requête diagnostic entre l'ECU et l'outil de diagnostic : écriture, lecture, suppression de données, etc., et il est identifié par un identifiant unique sous forme de code numérique hexadécimal :

Exemples :

- \$21** : Service de lecture des DIDs (Data by Identifier).
- \$2E** : Service d'écriture des DIDs.
- \$19** : Service de lecture des DTCs (Diagnostic Trouble Code).
- \$31** : Service pour routine.
- \$11** : Service pour reset de l'ECU.
- \$27** : Service pour déverrouillage du calculateur.
- \$10** : Service pour changer de session de diagnostic.

Avec l'augmentation constante des données échangées dans les véhicules modernes, notamment pour les mises à jour logicielles, les systèmes ADAS et les fonctionnalités connectées, le bus CAN atteint ses limites. C'est dans ce contexte qu'est apparu DoIP (Diagnostics over IP) norme ISO 13400, c'une extension du protocole UDS utilisant le réseau Ethernet/TCP-IP.

Contrairement au DoCAN (Diagnostics over CAN), qui repose sur un débit limité (1 Mbps), DoIP encapsule les services UDS dans des trames IP/TCP, offrant des vitesses de communication beaucoup plus élevées (jusqu'à 1 Gbps). Cela permet, de réduire considérablement le temps de flashage des calculateurs (de 40 minutes à moins de 5 minutes). DoIP ouvre également la voie à de nouvelles possibilités comme :

- Les mises à jour logicielles à distance OTA (Over-the-air).

- Les diagnostics multi-véhicules en usine ou à distance.
- La connexion directe à un ordinateur via un port Ethernet.
- Des communications sécurisées avec chiffrement TLS (Transport Layer Security)[6].

2.3.6 L'évolution du nombre de calculateur chez Stellantis

Les fabricants de voitures ont ajusté leur production à la demande des consommateurs en améliorant le confort des automobiles. Ainsi, les fabricants ont considérablement augmenté le nombre d'équipements des véhicules. Cette expansion des fonctionnalités n'est pas sans restrictions. Elle provoque l'expansion du nombre de calculateurs, l'accroissement des communications entre eux et une consommation d'énergie de plus en plus grande.

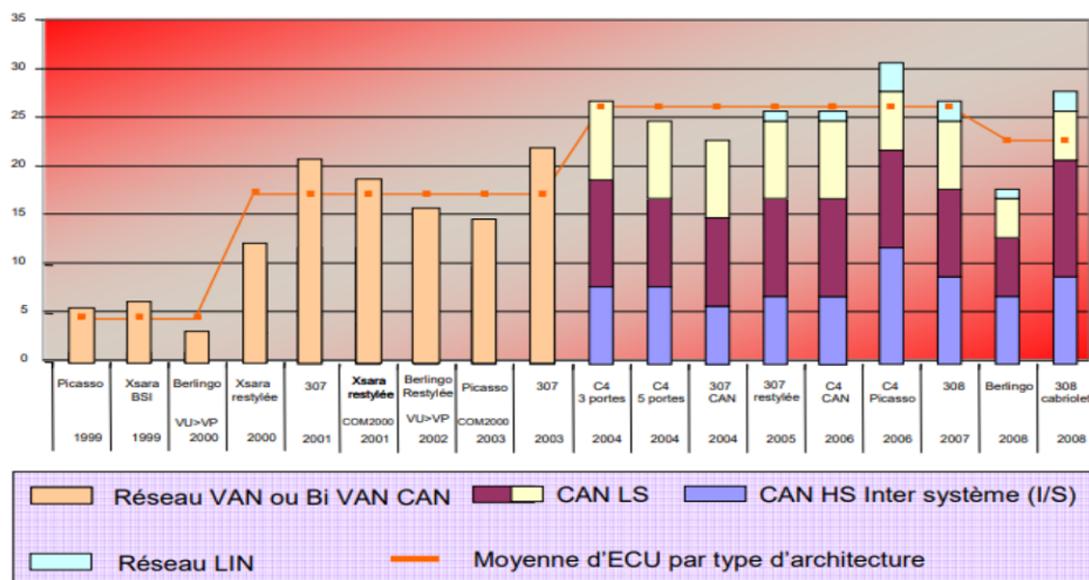


FIGURE 2.9 – L'évolution du nombre de calculateur

2.3.7 L'évolution des architectures chez Stellantis

Le terme « architecture » désigne la conception de systèmes organisés selon une structure cohérente et hiérarchisée. Chez Stellantis, une architecture fait référence à l'infrastructure globale du véhicule, qui fournit les canaux de communication nécessaires pour interconnecter l'ensemble des sous-domaines fonctionnels tels que le GMP

(Groupe Moto-Propulseur), le LAS (ADAS et systèmes d'aide à la conduite), la carrosserie (Body), le confort, l'InfoDiv (infodivertissement) et l'IHM (interface homme-machine).

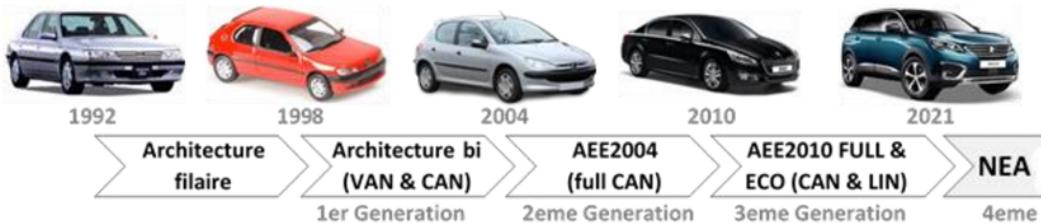


FIGURE 2.10 – Les générations des architectures chez Stellantis

L'architecture électronique des véhicules a connu une véritable révolution grâce à l'intégration des nouvelles technologies de communication interne entre les organes du véhicule.

1ère génération – Architecture bi VAN-CAN (1998) : Introduction généralisée du multiplexage via les protocoles VAN (Vehicle Area Network) et CAN HS, permettant une communication plus efficace entre les calculateurs.

2ème génération – AEE 2004 : Cette génération marque une rupture technologique avec l'abandon progressif du VAN au profit de la standardisation du protocole CAN, dans une architecture connue sous le nom de AEE 2004. Bien que plus robuste, cette architecture a montré ses limites face à la complexité croissante des nouveaux projets.

3ème génération – AEE 2010 : Face à la saturation de l'architecture précédente, la génération AEE 2010 a été introduite. Elle se décline en plusieurs évolutions : R1 (2010), R2 (2016) et R3 (2017). Une version optimisée, AEE 2010 ECO, accompagne cette évolution. Elle vise à réduire le nombre de réseaux, fusionner plusieurs fonctions dans un seul calculateur, et minimiser la taille et la complexité des autres composants.

Le groupe Stellantis développe une nouvelle architecture électrique de **4ème génération**, nommée NEA (New Electronic Architecture), qui joue le rôle de véritable système nerveux du véhicule connecté et autonome. Conçue pour répondre aux exigences de sécurité et de fiabilité des voitures autonomes, la NEA intègre des technologies de connectivité à distance via le système OTA. Cette connectivité permet aux utilisateurs d'effectuer à distance certaines opérations de maintenance, de mettre à jour ou enrichir les fonctionnalités du véhicule, et d'accéder à de nouveaux services, sans se rendre en concession et en toute sécurité.

2.3.8 Exemple d'une architecture AEE2010

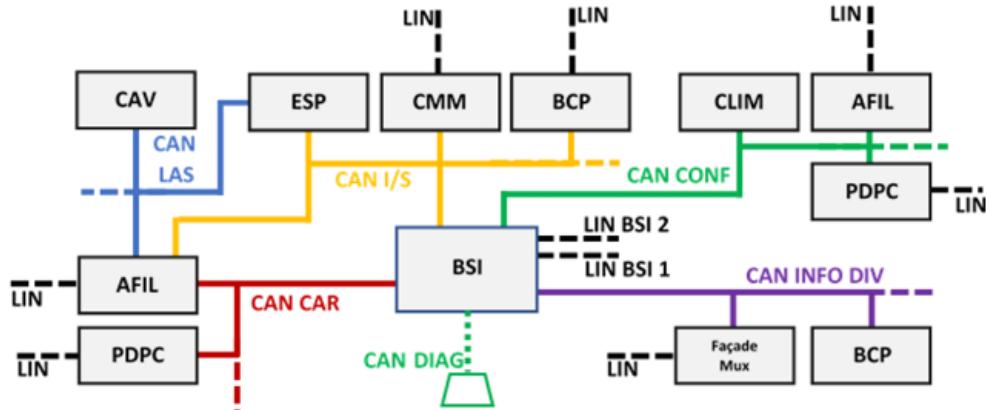


FIGURE 2.11 – Exemple d'une architecture AEE2010

L'Architecture Électrique et Électronique 2010 (AEE2010) est équipée de 6 réseaux CAN et de plusieurs réseaux LIN :

Le réseau '**CAN DIAG** (CAN Diagnostic) de 500Kbit/s jusqu'à 1Mbit/s : c'est la passerelle qui permet de connecter l'interface de diagnostic avec tous les réseaux à travers le calculateur central BSI ;

Les réseaux **CAN HS** de 500 Kbits / seconde :

- Réseau CAN I/S (inter-système) : pour l'échange de données critiques entre calculateurs centraux ;
- Réseau CAN LAS (liaison au sol) : pour freinage, stabilité, ADAS, etc ;

Les réseaux **LIN** :

- LIN BSI 1 et BSI 2 ;
- LIN HDC (haut de Colonne de direction comme commandes volant...) ;
- LIN BCP (projecteurs ou modules de portes) ;
- LIN AFIL (Alerte de Franchissement Involontaire de Ligne) ;
- LIN PDPC (Platine de Porte Conducteur) ;
- LIN CMM (Calculateur Moteur Multifonction) ;
- LIN F MUX (Façade Multiplexée).

On remarque que AEE2010 se base sur CAN et LIN ;

2.3.9 BSI

Le Boîtier de Servitude Intelligent (BSI), en anglais Built-in System Interface, considéré comme le calculateur central du véhicule, constitue un élément clé de l'architecture électrique. Il se compose de deux sous-ensembles principaux :

- Une **partie électronique**, dédiée à la gestion et au pilotage des différentes fonctions embarquées du véhicule.
- Une **partie puissance**, chargée de la distribution et de la protection des alimentations électriques destinées aux organes du véhicule, en particulier ceux situés dans l'habitacle.

Il fournit les fonctionnalités système suivantes :

- Distribution et protection des alimentations électriques pour les composants de l'habitacle.
- Sauvegarde des anomalies signalées par les calculateurs d'habitacles.
- Obtention de entrées analogiques.
- Acquisition et pilotage des entrées/sorties Tout Ou Rien
- Diffusion de l'état du SEV (Système Électrique Véhicule) sur les réseaux multiplexés.
- Production du signal de réveil habitacle via une alimentation commutée (+ CAN).
- Production du signal de réveil I/S par une alimentation commutée RCD (Réveil Commandé à Distance).
- Passerelle entre différents réseaux (y compris le diagnostic).
- Diagnostic des fonctions internes au calculateur.

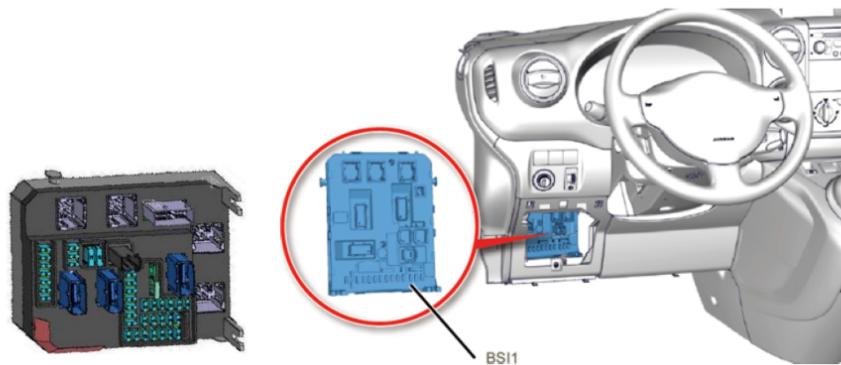


FIGURE 2.12 – Calculateur BSI et son emplacement dans les véhicules

Le BSI, en tant que nœud central interconnecté à plusieurs réseaux multiplexés, héberge de nombreuses fonctions véhicule, il permet également l'interaction entre ces

fonctions, favorisant la coordination entre différents sous-systèmes. En raison de sa position stratégique au sein de l'architecture électrique et de sa connexion à l'ensemble des réseaux multiplexés du véhicule, le BSI a également pour rôle l'hébergement du JDD (Journal des Défauts). Ce journal centralise les anomalies détectées, qu'elles soient locales au BSI ou externes. Le JDD BSI enregistre ainsi l'apparition et la disparition des défauts, assurant une traçabilité complète des dysfonctionnements électroniques du véhicule. Cette centralisation facilite le diagnostic en atelier, améliore le suivi qualité, et contribue à l'efficacité des processus de maintenance et de validation, notamment dans les environnements de test HIL où l'analyse fine des défauts est essentielle[7].

Le BSI est physiquement connecté aux réseaux :

CAN LS CAR (Carrosserie), CAN LS CONF (Confort), CAN LS INFODIV, CAN HS I/S, LIN BSI 1, LIN BSI 2.

2.4 Banc et environnement de test HIL

2.4.1 Test HIL

2.4.1.1 définition et généralités

Le test HIL représente une modalité de simulation en temps réel. L'intégration d'un élément tangible, désigné comme système de contrôle, dans la boucle explique pourquoi la simulation HIL se distingue des autres types de simulations en temps réel. Selon le contexte, le système de contrôle peut être qualifié d'« unité de contrôle électronique » s'il est utilisé dans les secteurs automobile ou aéronautique, ou encore de « contrôleur logique programmable » s'il est destiné à un domaine de contrôle des processus.

La machine ou la composante physique du dispositif (l'installation) est généralement relié au système de commande, par l'intermédiaire d'actionneurs et de capteurs. Grâce aux tests HIL, une simulation de l'installation, appelée « simulateur HIL », remplace l'usine. Si le simulateur HIL est correctement élaboré, il répliquera précisément l'équipement et pourra servir à tester le système de contrôle [8].

Le HIL test est une méthode qui connecte les signaux réels d'un contrôleur à un banc de test HIL, recréant la réalité et amenant le calculateur à croire qu'il est intégré au produit assemblé. Le processus de conception et de test se déroule comme s'il était basé sur un système du monde réel. Cela facilite l'exploration de milliers de scénarios

potentiels pour tester le contrôleur de manière adéquate sans les dépenses et le temps liés aux essais physiques concrets. L'environnement de test se compose d'un ordinateur de supervision hébergeant le système TestInVIEW pour la préparation, l'exécution et l'analyse des tests automatiques.

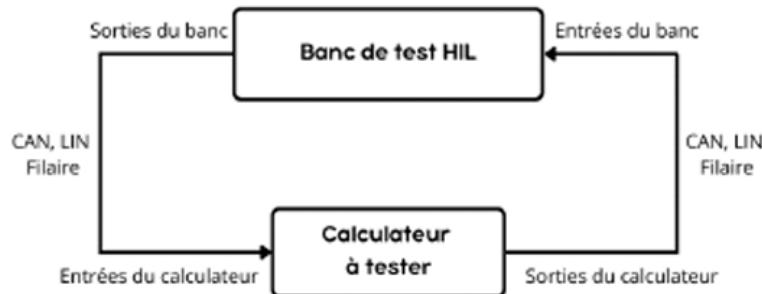


FIGURE 2.13 – Principe de Hardware in the loop

L'environnement de test employé pour vérifier le software embarqué du calculateur BSI se compose d'un PC de supervision qui héberge le système TestInView, utilisé pour préparer, lancer et gérer les tests automatiques.

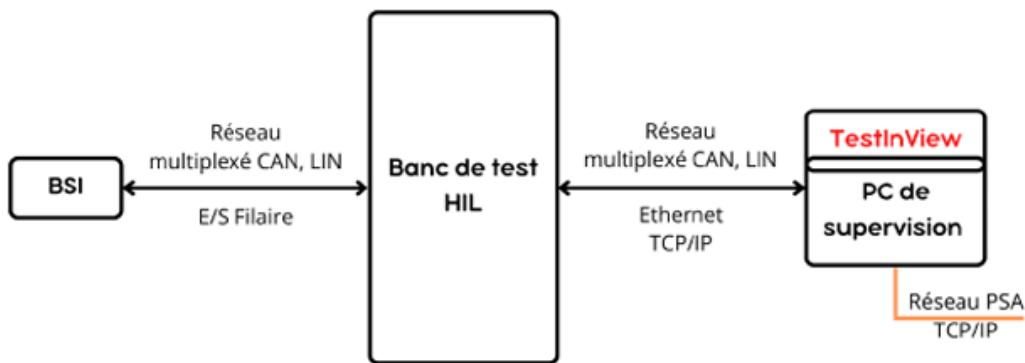


FIGURE 2.14 – Environnement du test du BSI

Le PC de supervision est connecté au banc HIL via une liaison TCP/IP (Transmission Control Protocol/Internet Protocol), et pour ce faire, nous utilisons une connexion à distance pour accéder au PC de supervision (PC du banc HIL). Le banc HIL est connecté au BSI via un réseau multiplexé (CAN et LIN), ainsi qu'à l'aide d'une connexion câblée (sous forme de signaux numériques et analogiques). L'objectif de ce banc d'essai est de simuler les signaux d'entrée du boîtier de servitude intelligent 'BSI', puis de comparer les signaux de sortie du boîtier avec les résultats prévus.

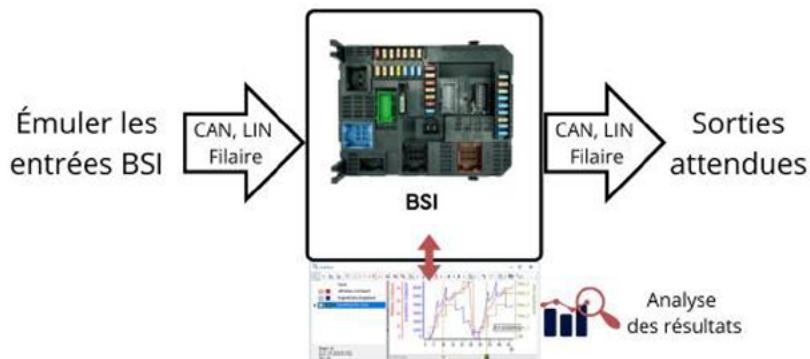


FIGURE 2.15 – BSI sous test

Le banc HIL est une sorte de plateforme qui a pour but d'offrir au BSI un cadre d'essai se rapprochant de celui d'un véritable véhicule, cela implique la reproduction intégrale des éléments d'un véhicule réel afin que le BSI puisse se sentir dans un cadre authentique et non pas dans une simulation. On peut affirmer qu'à l'aide du banc HIL, il est possible de transmettre des données (trames CAN et LIN) du BSI vers les autres unités de contrôle simulées, et même de recevoir des trames en provenance de ces dernières. De ce fait, la simulation de l'échange d'informations pourra être effectuée.



FIGURE 2.16 – Banc de test HIL

Ce simulateur est donc capable de reproduire précisément le système contrôlé et sa dynamique, ainsi que son équipement (capteurs/actionneurs), afin d'expérimenter leurs

interactions en boucle fermée, sans avoir à recourir à un système réel. En plus de ces caractéristiques, la simulation HIL remédie efficacement aux insuffisances des méthodes classiques : en minimisant les risques, les dépenses et le laps de temps requis pour l'essai des systèmes embarqués compliqués, elle se positionne comme norme pour un grand nombre de secteurs industriels à travers le monde, y compris l'industrie automobile.

2.5 Conclusion

Ce chapitre a présenté un aperçu global des principaux composants et technologies des systèmes automobiles modernes. Des différents types de véhicules électrifiés aux architectures électriques et électroniques, en passant par les protocoles de communication comme LIN et CAN, nous avons mis en lumière les fondations techniques sur lesquelles repose un véhicule actuel. Le rôle central du BSI et l'intérêt des tests HIL ont également été soulignés.

CHAPITRE 3

CONCEPTION ET RÉALISATION DE L'APPLICATION

3.1 Introduction

Ce chapitre décrit la démarche adoptée pour concevoir et développer l'application. Il présente les choix de modélisation UML, la structure de la base de données, les outils techniques utilisés, ainsi que l'implémentation des différentes interfaces de l'application.

3.2 Conception de l'application

3.2.1 Langage de modélisation UML

Le langage UML (Unified Modeling Language, ou langage de modélisation unifié) a été pensé pour être un langage de modélisation visuelle commun, et riche sémantiquement et syntaxiquement. Il est destiné à l'architecture, la conception et la mise en œuvre de systèmes logiciels complexes par leur structure aussi bien que leur comportement. L'UML a des applications qui vont au-delà du développement logiciel, notamment pour les flux de processus dans l'industrie.

Il ressemble aux plans utilisés dans d'autres domaines et se compose de différents types de diagrammes. Dans l'ensemble, les diagrammes UML décrivent la limite, la structure et le comportement du système et des objets qui s'y trouvent.

L'UML n'est pas un langage de programmation, mais il existe des outils qui peuvent

être utilisés pour générer du code en plusieurs langages à partir de diagrammes UML. L'UML a une relation directe avec l'analyse et la conception orientées objet.



FIGURE 3.1 – Logo d'UML

L'UML est utilisé pour représenter visuellement des systèmes informatiques. Il permet de :

- Faciliter la communication et la compréhension entre les parties prenantes en fournissant un langage commun.
- Documenter les différentes parties d'un système informatique pour une meilleure compréhension et maintenance.
- Concevoir des systèmes informatiques de manière plus efficace en planifiant et organisant les différentes parties du système.

3.2.2 Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation peut servir à résumer les informations des utilisateurs de votre système (également appelés acteurs) et leurs interactions avec ce dernier. La création de ce type de diagramme UML requiert un ensemble de symboles et de connecteurs spécifiques. Lorsqu'ils sont bien conçus, les diagrammes de cas d'utilisation peuvent aider votre équipe à collaborer et représenter :

- Les scénarios dans lesquels votre système ou application interagit avec des personnes, des organisations ou des systèmes externes.
- Les objectifs que votre système ou application permet aux entités (appelées acteurs) d'atteindre.
- La portée de votre système.

Le diagramme de cas d'utilisation se compose de trois éléments principaux : Un Acteur, Un cas d'utilisation et Les relations.

Acteur principal Valideur HIL : c'est l'utilisateur principal de l'application. Il peut insérer, comparer, consulter, supprimer des exigences, vérifier l'applicabilité avec les projets et également gérer la base de données.

Acteur secondaire Admin : L'Admin intervient ponctuellement dans le système pour des actions de contrôle ou de validation. Son rôle principal est de valider les demandes sensibles, notamment la suppression de projets. Cette validation permet de garantir la sécurité et la traçabilité.

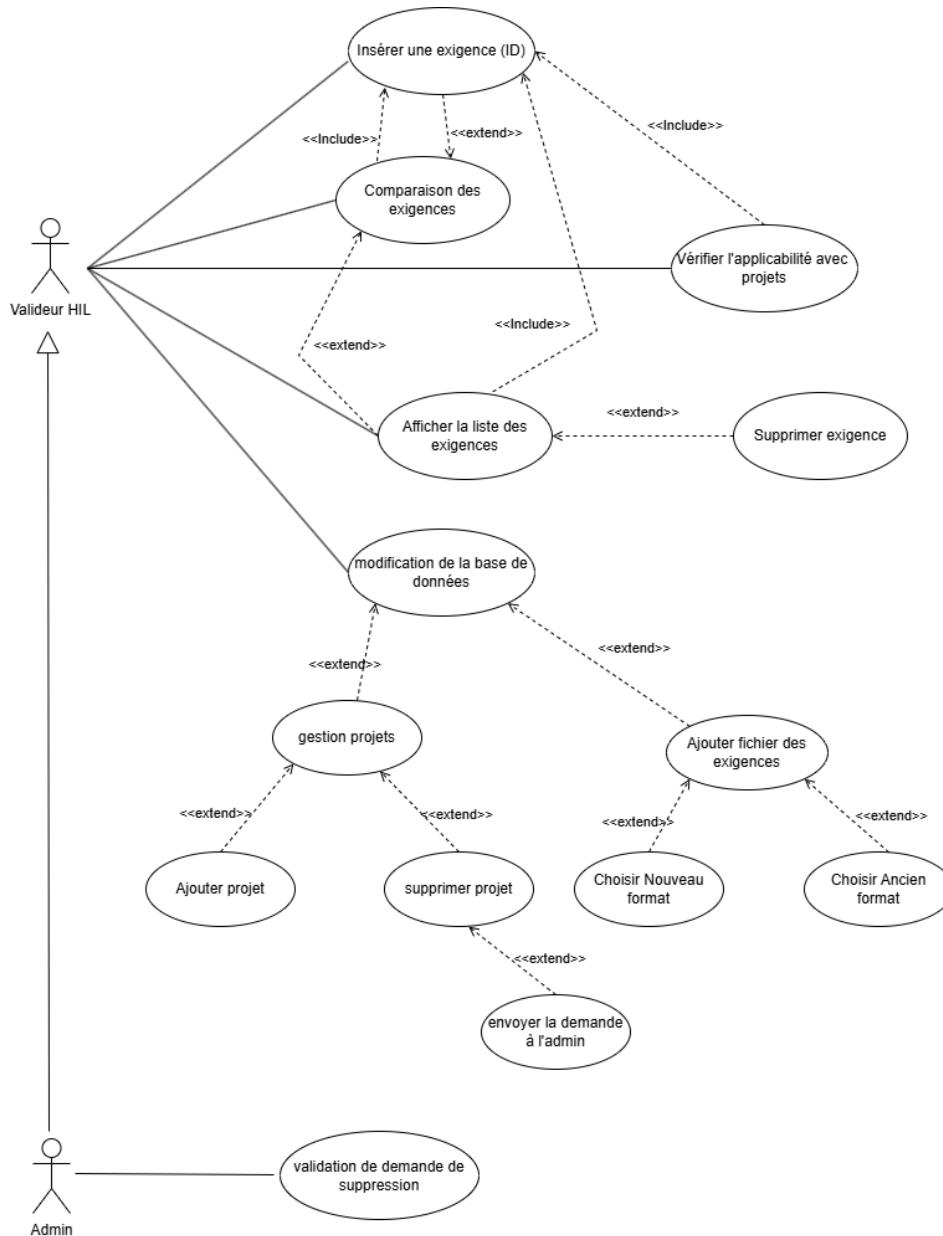


FIGURE 3.2 – Diagramme de cas d'utilisation

Insérer une exigence (ID) : Est une action de départ obligatoire. Le valideur HIL commence par insérer une exigence en saisissant son ID. C'est une condition préalable à d'autres fonctionnalités comme la comparaison ou la vérification d'applicabilité.

Comparaison des exigences : Cette action est accessible uniquement après l'insertion d'au moins une exigence. Elle Permet de comparer plusieurs exigences déjà insérées.

Depuis cette fonctionnalité, le valideur HIL peut revenir à l'insertion d'exigences pour en ajouter d'autres et il peut aussi afficher la liste des exigences pour consulter les exigences déjà insérées.

Afficher la liste des exigences : Cette action permet de visualiser les exigences insérées et offre la possibilité de supprimer une exigence à tout moment sans impacter le résultat précédent de la comparaison.

Vérifier l'applicabilité avec projets : Cette étape permet de savoir si les exigences sont applicables à des projets spécifiques. Elle nécessite qu'au moins une exigence soit déjà insérée.

Modification de la base de données : L'application permet au valideur HIL d'accéder à des fonctions avancées de gestion via la modification de la base de données. Ce cas d'utilisation offre la possibilité de gérer les projets (ajouter ou supprimer un projet, par exemple en cas de nouvelle version), ainsi que d'ajouter de nouveaux fichiers d'exigences.

Validation de la demande de suppression (Admin) : L'administrateur joue un rôle de contrôle dans le processus de gestion des projets. Lorsqu'un valideur HIL souhaite supprimer un projet, il doit d'abord envoyer une demande de suppression. Cette demande est ensuite soumise à l'Admin, qui est le seul acteur habilité à la valider. Cette action de validation permet de sécuriser les opérations sensibles et garantir ainsi la traçabilité.

3.2.3 Diagramme de classe

Les diagrammes de classes font partie de six types de diagramme structurel. Ces diagrammes sont essentiels à la structure statique d'un système. Selon la complexité de ce dernier, on peut utiliser un ou plusieurs diagrammes de classes pour le modéliser complètement.

Un diagramme de classes sert à donner une vue statique générale d'un système, en mettant l'accent sur les classes, leurs relations et leurs attributs/méthodes. Absolument, un diagramme de classes se compose des éléments ci-dessous :

- Classes : les classes à de forme de rectangles avec leurs noms à l'intérieur. Elles

peuvent contenir des attributs (I.e. variables) et des méthodes (I.e. fonctions). Alors qu'une classe peut se décomposer en trois parties (trois sous rectangles), la première pour le nom, la deuxième pour les attributs et la troisième pour les méthodes.

- Attributs : les attributs sont les variables qui dépeignent les caractéristiques d'une classe. Ils sont représentés dans la deuxième partie comme on a déjà indiqué, avec leur type de donnée. Par exemple, un attribut "nom" de type "string".
- Méthodes : il s'agit des fonctions qui définissent le comportement d'une classe. Elles sont représentées dans la troisième partie, avec leurs noms, leurs paramètres et leur type de retour.
- Associations : les associations représentent les relations qui existent entre les classes. Ils montrent la façon dont les instances des classes sont liées les unes aux autres. Il peut y avoir des associations à sens unique ou à double sens.
Parmi les associations, on distingue aussi deux types particuliers :
 - **Agrégation** : une relation indiquant qu'une classe contient une autre classe, mais sans lien de dépendance forte. Elle est représentée par un losange creux.
 - **Composition** : une relation plus forte où la classe contenue ne peut pas exister sans la classe contenante. Elle est représentée par un losange noir.
- Héritage : l'héritage permet de définir une nouvelle classe, nommée classe dérivée, de celle qui existe déjà. La nouvelle classe hérite des attributs et des méthodes de la classe parente.

Le diagramme de classe est fréquemment utilisé dans la modélisation et la conception des applications pour saisir les concepts clés et faciliter la communication entre les membres de l'équipe de développement.

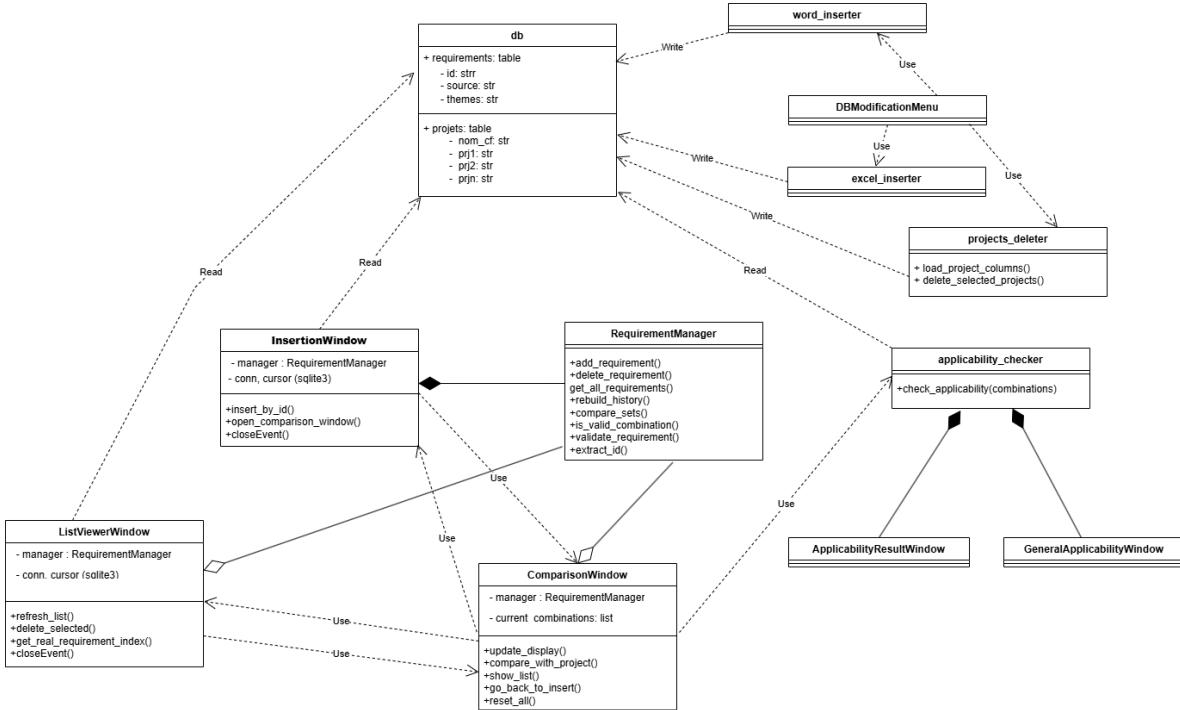


FIGURE 3.3 – Diagramme de classe de l'application

3.2.4 Diagramme des séquences

Un diagramme de séquence constitue un autre type parmi les types de diagrammes UML considéré comme un outil visuel soutien à comprendre l'enchaînement d'un processus dans un système.

Il sert à illustrer visuellement l'interaction entre les acteurs ou les objets d'un système en ordre chronologique.

Tandis qu'un diagramme de séquence montre la manière dont les éléments du système interagissent les uns avec les autres et avec les acteurs en indiquant l'ordre des messages échangés entre eux au cours de l'exécution d'un scénario spécifique.

Les principaux éléments d'un diagramme de séquence comprennent :

- Les acteurs ou les objets : ils représentent les éléments qui inter agissent ensemble dans le système.
- Les messages : ils représentent les échanges d'information entre les objets. Ces messages peuvent s'agir d'appels de méthodes, de signaux, de demandes d'intervention, etc.
- Les activations : ils indiquent la durée de fonctionnement d'un objet ou de traitement d'un message.

- Les lignes de vie : ils représentent le cycle de vie des objets impliqués dans l'interaction, indiquant quand ils sont actifs et quand ils sont inactifs.

Grâce à quatre éléments, la représentation graphique offert par le diagramme de séquence sera claire et séquentielle des corrélations entre les objets, permettant à comprendre la façon dont un processus ou un scénario fonctionne au sein d'un système.

Tant que le système à modéliser n'est pas extrêmement simple, on ne peut pas modéliser la dynamique générale du système en un seul diagramme. En conséquence, nous utiliserons un ensemble des diagrammes de séquences chacun correspondant à une sous mission du système, en général pour illustrer une situation d'utilisation.

Nous avons mis au point cinq diagrammes de séquences :

3.2.4.1 Diagramme de séquence d'insertion des exigences

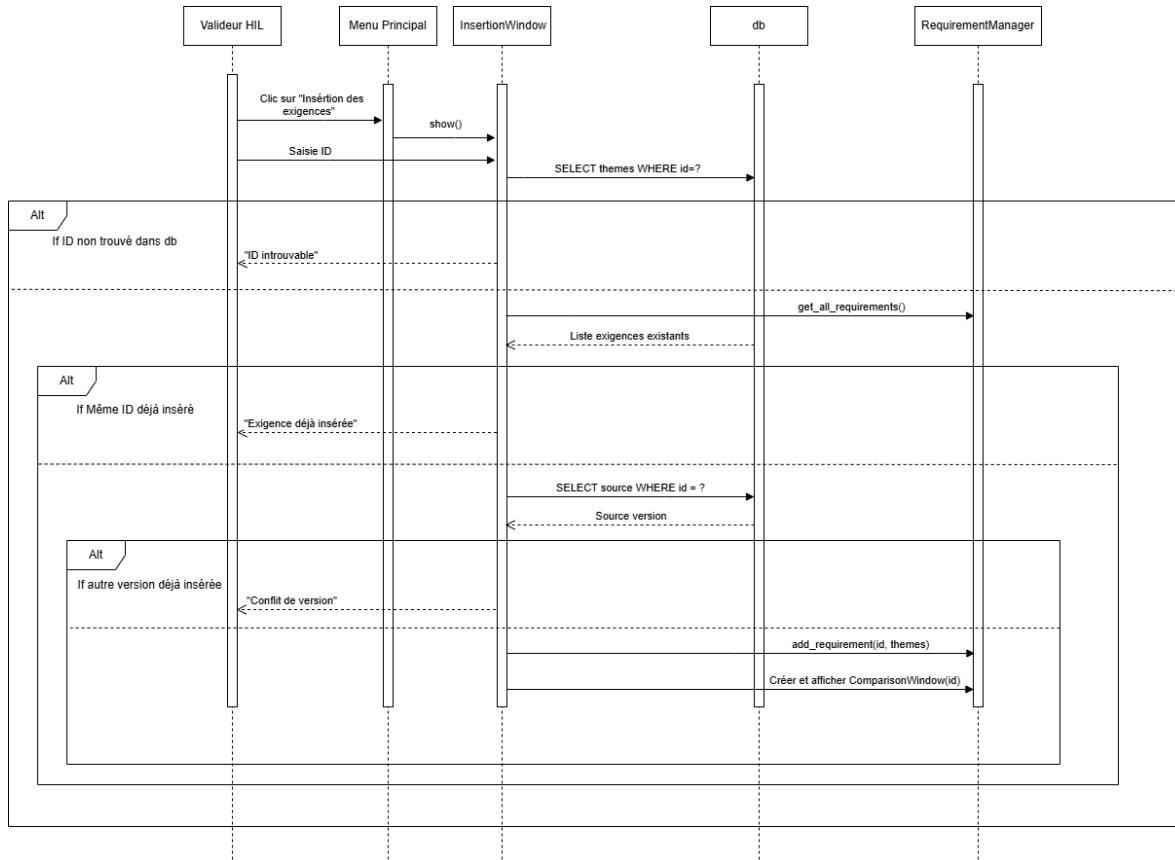


FIGURE 3.4 – Diagramme de séquence d'insertion des exigences

Lorsque le valideur HIL clique sur le bouton « *Insérer des exigences* » depuis le menu principal, une fenêtre s'affiche qui permet à l'utilisateur de saisir l'ID d'une exi-

gence, l'application vérifie d'abord si l'ID saisi correspond à une exigence présente dans la base de données. Si l'ID est introuvable, un message d'erreur « *ID introuvable* » est affiché et l'opération est interrompue. Dans le cas où l'ID est trouvé, l'application récupère la liste des exigences déjà insérées afin de vérifier si cette exigence a été précédemment ajoutée. Si le même ID avec les mêmes données a déjà été inséré, le système informe l'utilisateur via le message « *Exigence déjà insérée* ». Si une autre version de la même exigence (par exemple, issue d'un autre fichier source) est détectée, un message de « *Conflit de version* » est affiché, et l'insertion est également bloquée. En l'absence de conflit ou de doublon, l'exigence est ajoutée à la base via la fonction `add_requirement()`, et la fenêtre de comparaison est automatiquement générée pour permettre l'analyse des exigences.

3.2.4.2 Diagramme de séquence de comparaison des exigences

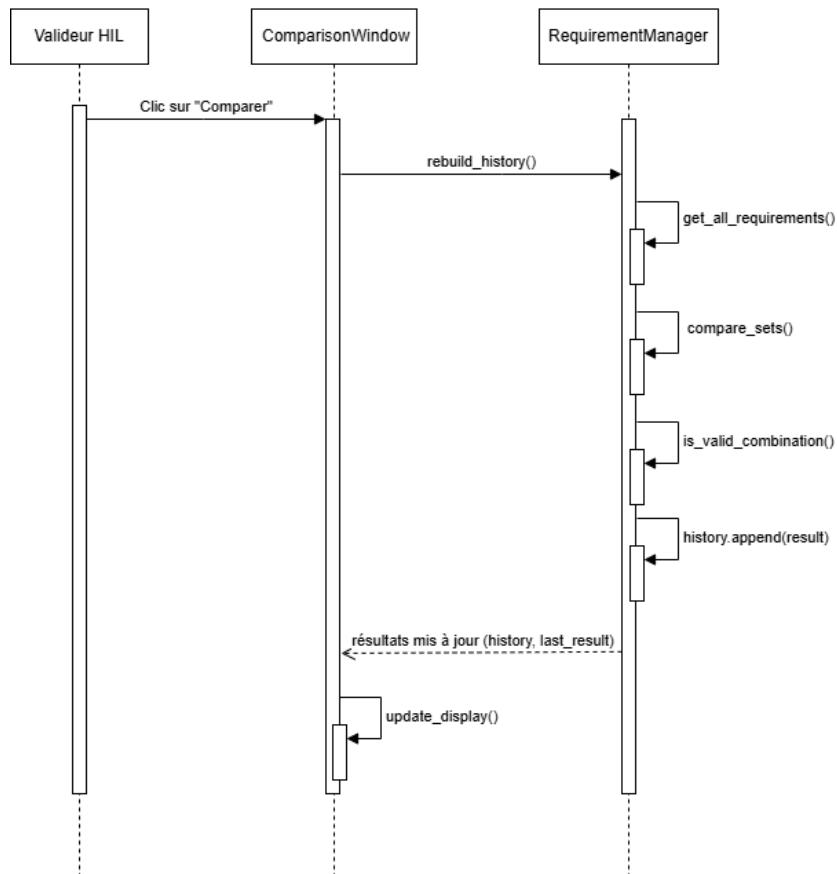


FIGURE 3.5 – Diagramme de séquence de comparaison des exigences

Lorsque l'utilisateur clique sur le bouton « *Comparer* » dans la fenêtre de comparaison, le processus de comparaison des exigences est déclenché. L'interface appelle la méthode `rebuild_history()` afin de reconstruire l'historique des comparaisons précédentes.

Cette méthode commence par récupérer l'ensemble des exigences insérées via la fonction `get_all_requirements()` du module `RequirementManager`. Ensuite, la comparaison est effectuée en générant des paires d'exigences grâce à la fonction `compare_sets()`. Chaque paire ou combinaison est ensuite soumise à la fonction `is_valid_combination()` qui filtre les combinaisons non valides selon des règles prédéfinies.

Les résultats valides sont ajoutés à l'historique à l'aide de `history.append(result)`, et le dernier résultat est mis à jour. Enfin, l'interface `ComparisonWindow` utilise la méthode `update_display()` pour afficher à l'utilisateur la comparaison actuelle ainsi que l'historique mis à jour.

3.2.4.3 Diagramme de séquence de la liste des exigences

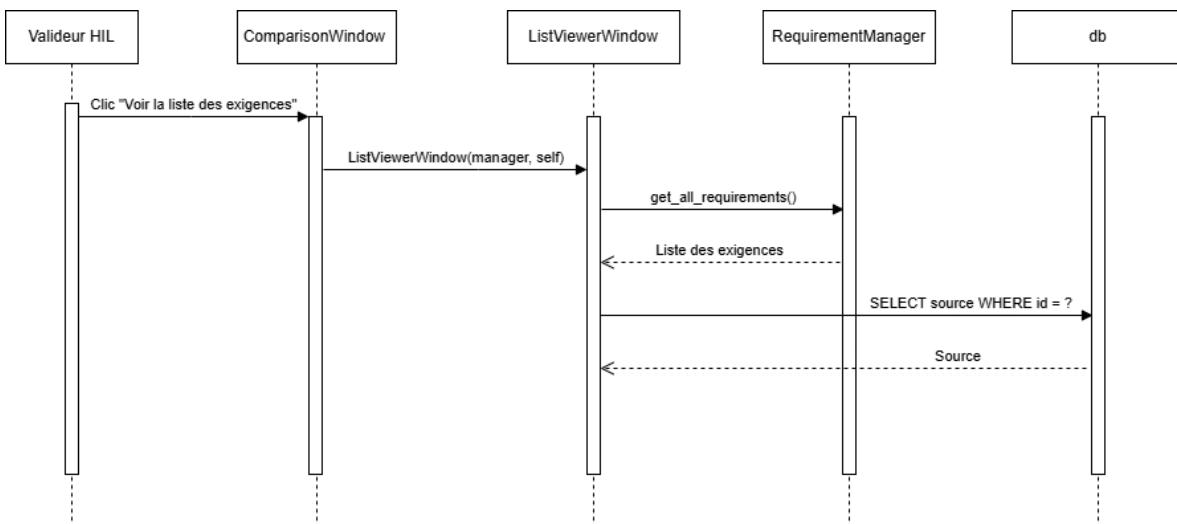


FIGURE 3.6 – Diagramme de séquence la liste des exigences

Lorsque le valideur HIL clique sur le bouton « *Voir la liste des exigences* » depuis la fenêtre de comparaison, une nouvelle fenêtre `ListViewerWindow` est créée.

À l'ouverture, `ListViewerWindow` appelle la méthode `get_all_requirements()` du module `RequirementManager`, qui récupère depuis la base de données l'ensemble des exigences actuellement insérées dans l'application.

Pour chaque exigence, une requête supplémentaire de type `SELECT source WHERE id = ?` est effectuée afin de récupérer la source exacte (le fichier d'origine). Une fois ces

informations obtenues, la liste complète des exigences, accompagnée de leurs sources, est affichée à l'utilisateur sous une forme structurée.

Ce scénario permet ainsi au validateur de consulter toutes les exigences insérées, d'en vérifier l'origine, et éventuellement de décider de les supprimer ou non, selon le besoin.

3.2.4.4 Diagramme de séquence de suppression d'une exigence

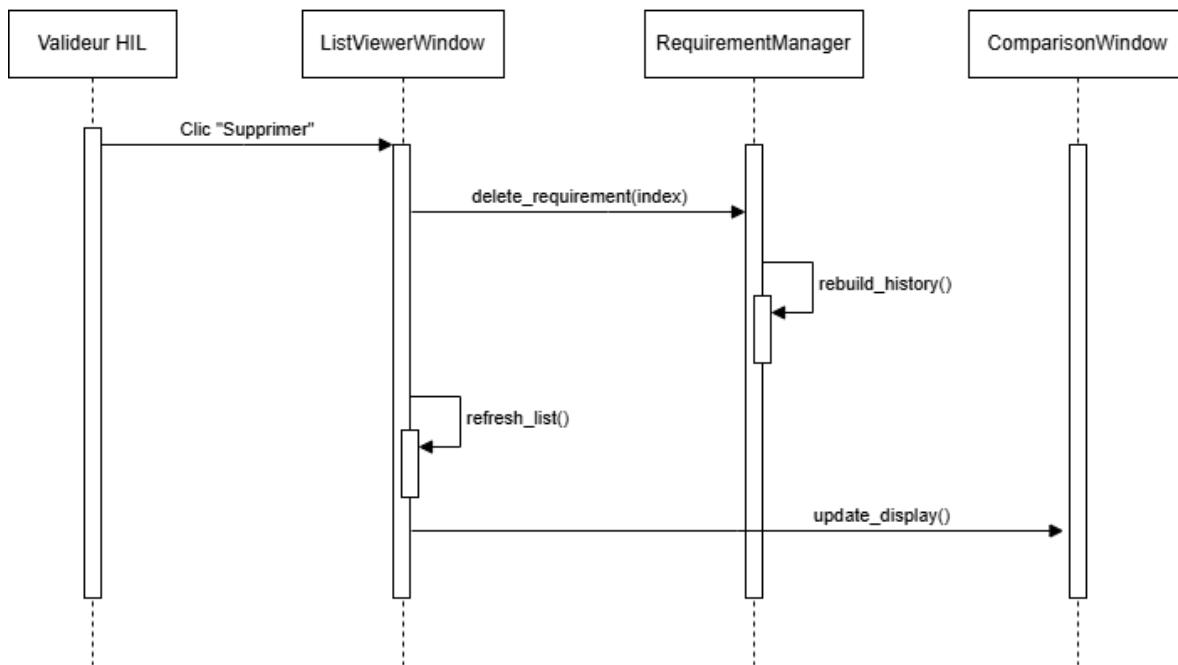


FIGURE 3.7 – Diagramme de séquence de suppression d'une exigence

Lorsque l'utilisateur clique sur le bouton « Supprimer » depuis la fenêtre de la liste des exigences, la méthode `delete_requirement(index)` est appelée. Cette méthode transmet l'index de l'exigence à supprimer au module `RequirementManager`, qui se charge de retirer l'exigence correspondante de la base active.

Après suppression, on appelle la fonction `rebuild_history()` afin de reconstruire l'historique des comparaisons, pour garantir la cohérence avec les nouvelles exigences restantes. Une fois l'historique recalculé, la méthode `update_display()` est invoquée pour mettre à jour la fenêtre `ComparisonWindow`, assurant ainsi que l'interface reflète les données actualisées.

En parallèle, la liste visible par l'utilisateur dans `ListViewerWindow` est rafraîchie via l'appel à `refresh_list()`, pour ne plus afficher l'exigence supprimée. Ce scénario assure une suppression dynamique et cohérente sans impacter négativement l'état global du système.

3.2.4.5 Diagramme de séquence de vérification d'applicabilité avec projets

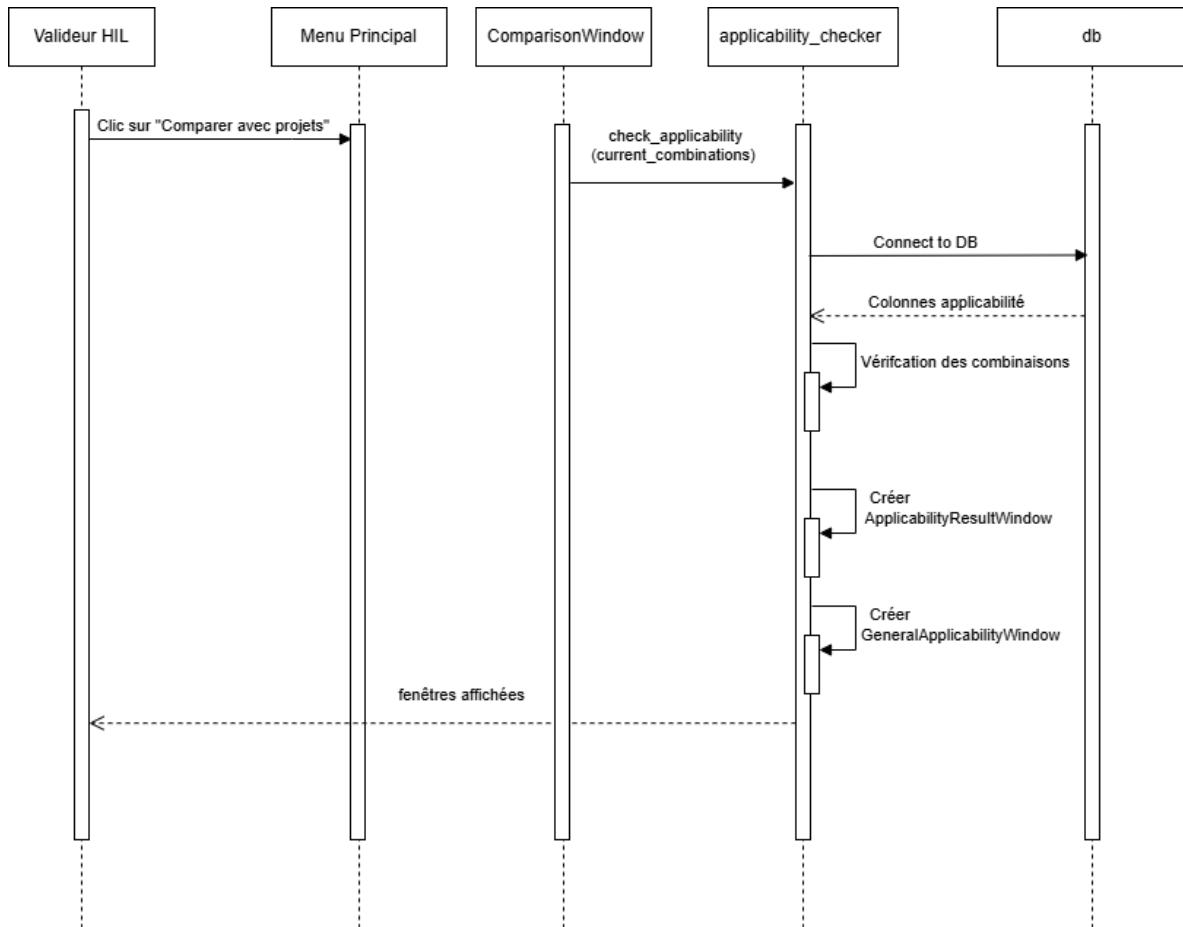


FIGURE 3.8 – Diagramme de séquence de vérification d'applicabilité avec projets

Lorsque le valideur HIL clique sur le bouton « *Comparer avec projets* » depuis le menu principal, la fenêtre **ComparisonWindow** transmet les combinaisons d'exigences actuellement affichées à la fonction `check_applicability(current_combinations)` du module **applicability_checker**.

Ce module commence par se connecter à la base de données afin de récupérer les colonnes liées à l'applicabilité des projets (`S`, `0`, `nan`, ou `NULL`). Une fois ces données obtenues, le module procède à une vérification de chaque combinaison pour déterminer si elle est applicable selon les règles définies. Une combinaison est considérée comme applicable si toutes ses thématiques sont marquées comme sélectionnées ou optionnelles dans la base.

Après l'analyse, deux interfaces sont générées et affichées :

- `ApplicabilityResultWindow` : qui affiche les combinaisons applicables, non applicables et non existantes de manière différenciée.
- `GeneralApplicabilityWindow` : qui présente une vue globale des projets et de leur couverture par les exigences analysées.

3.2.4.6 Diagramme de séquence de Modification de la base de données

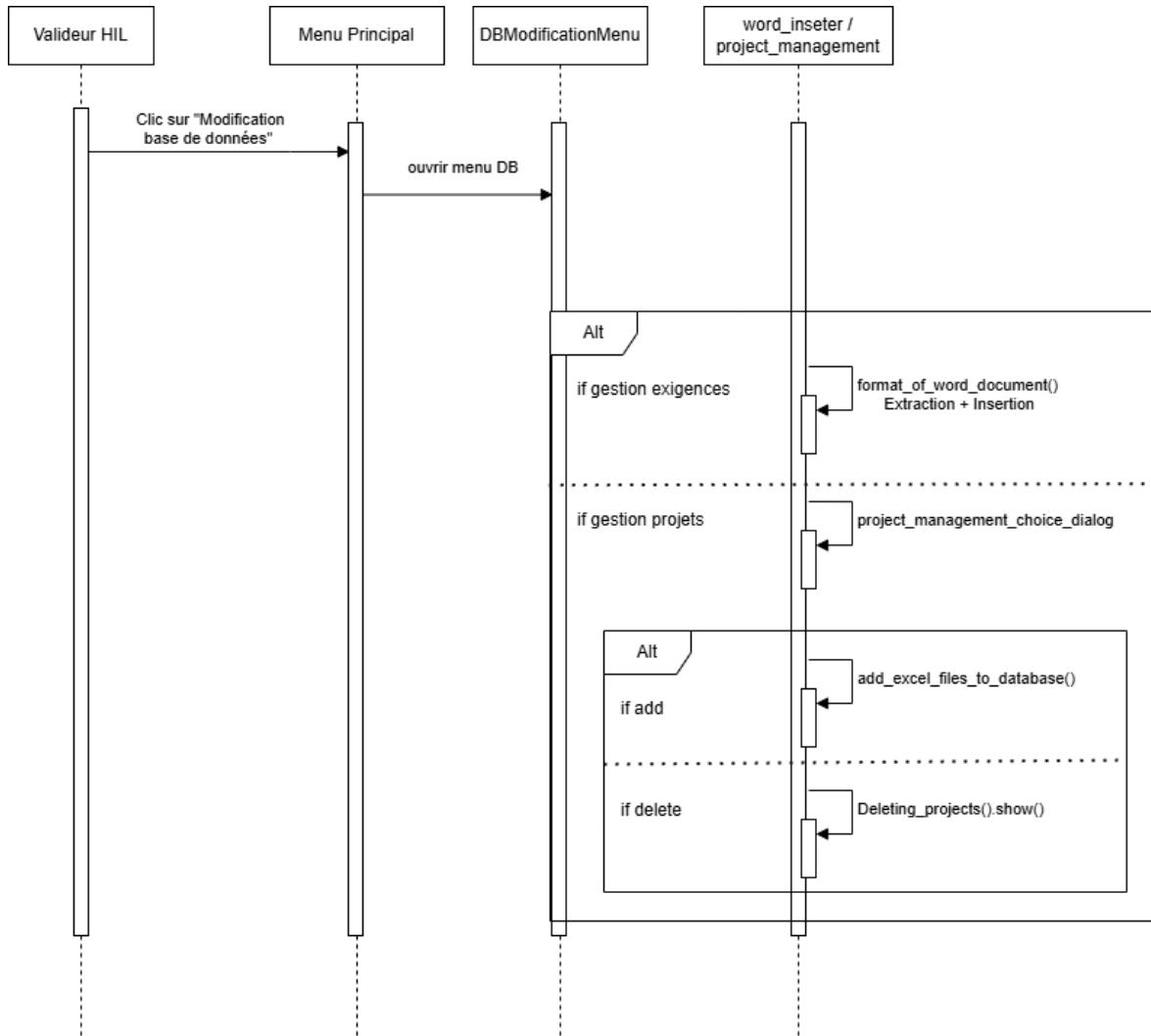


FIGURE 3.9 – Diagramme de séquence de Modification de la base de données

Lorsque le valideur HIL sélectionne l'option « *Modification de base de données* » depuis le menu principal, la fenêtre `DBModificationMenu` est affichée. À partir de cette fenêtre, deux possibilités sont proposées à l'utilisateur : la gestion des exigences ou la gestion des projets. Ce choix est représenté par un bloc `alt` dans le diagramme.

Dans le premier cas, si l'utilisateur choisit « *Gestion exigences* », la méthode `format_of_word_document()` du module `word_inserter` est appelée. Celle-ci permet à l'utilisateur de sélectionner un fichier Word, d'en préciser le format (ancien ou nouveau), puis d'extraire et d'insérer automatiquement les exigences contenues dans ce fichier dans la base de données.

Dans le second cas, si l'utilisateur choisit « *Gestion projets* », une boîte de dialogue `ProjectManagementChoiceDialog` est affichée pour lui permettre de choisir entre deux actions : « *ajouter un projet* » ou « *supprimer un projet* ». Ce choix est modélisé par un second bloc `alt` imbriqué. Si l'utilisateur choisit d'ajouter, la méthode `add_excel_files_to_database()` du module `excel_inserter` est appelée pour importer des projets à partir de fichiers Excel. Si l'utilisateur opte pour la suppression, une instance de la fenêtre `Deleting_projects()` est affichée, permettant de sélectionner et supprimer un ou plusieurs projets déjà présents dans la base.

3.2.4.7 Diagramme de Confirmation administrateur avant suppression

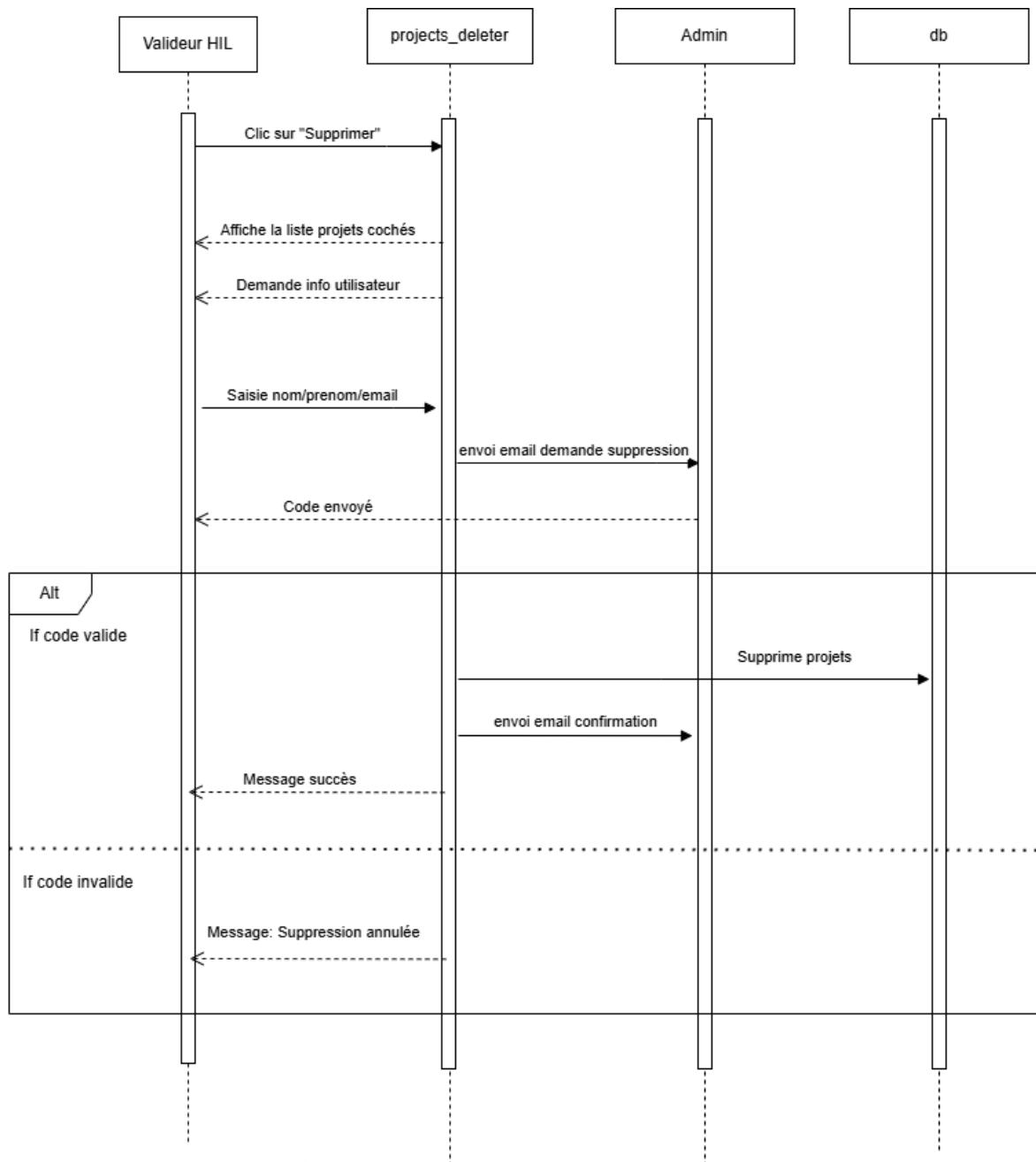


FIGURE 3.10 – Diagramme de Confirmation administrateur avant suppression

Lorsqu'un validateur HIL sélectionne des projets à supprimer, une demande d'information est lancée pour saisir son nom, prénom et adresse email. Un email est alors automatiquement envoyé à l'administrateur pour signaler cette demande. Une fois le

code de validation reçu, l'utilisateur le saisit dans l'interface. Si le code est correct, la suppression des projets est effectuée dans la base de données et un email de confirmation est renvoyé à l'administrateur. En cas de code invalide, la suppression est annulée.

3.3 Conception de la base de données

La base de données utilisée dans ce projet repose sur une structure simple composée de deux entités principales : requirements et projects. Ces deux tables sont indépendantes, car elles répondent à des finalités différentes dans l'architecture du système.

La table requirements centralise les exigences extraites automatiquement depuis les documents Word. Chaque enregistrement contient l'identifiant de l'exigence, les thématiques extraites, la source du document, la date d'insertion (timestamp), et le nombre d'occurrences détectées. Cette table constitue la base de référence pour l'analyse logique, les comparaisons entre exigences et la traçabilité.

La table projects contient une liste de thématiques (nom_cf) associées à différents projets. Chaque colonne de cette table correspond à un projet spécifique, et les cellules indiquent si la thématique est sélectionnée (S), optionnelle (O), non applicable (nan), ou non existante (NULL). Cette table est utilisée pour vérifier l'applicabilité des exigences extraites dans un contexte projet donné.



global.db		Rows: 6,905				Upgrade to PRO	
TABLES		id	themes	timestamp	source	occurrences	#
		1	REQ-05			2	
		2	REQ-05			2	
		3	REQ-05			2	
		4	REQ-05			2	
		5	REQ-05			2	
		6	REQ-05			2	
		7	REQ-05			2	
		8	REQ-05			2	
		9	REQ-05			2	
		10	REQ-05			2	
		11	REQ-05			1	
		12	REQ-05			2	
		13	REQ-05			2	
		14	REQ-05			2	
		15	REQ-05			2	
		16	REQ-05			2	
		17	REQ-05			2	
		18	REQ-05			2	
+ 6,906							

FIGURE 3.11 – Table des exigences

3.3. CONCEPTION DE LA BASE DE DONNÉES

FIGURE 3.12 – Table des projets

3.3.1 Outils utilisés

3.3.1.1 Langage principal Python 3

Afin de sélectionner le langage de programmation le plus adapté à notre projet, un benchmarking a été réalisé entre plusieurs langages envisageables : Python, VBA, et C. Le tableau suivant résume les avantages et limitations de chacun selon des critères pertinents pour les objectifs du projet.

Critère	Python	VBA	C
Courbe d'apprentissage	Facile, syntaxe intuitive et accessible.	Simple pour les utilisateurs d'Excel.	Difficile, syntaxe bas niveau.
Interface graphique	Excellente avec PyQt5 ou Tkinter.	Très limitée (UserForms).	Implémentation manuelle (GTK, WinAPI...).
Connexion base de données	Simple (SQLite, SQLAlchemy).	Possible mais peu optimisé (via ADO).	Requiert gestion manuelle (pointeurs, drivers).
Portabilité multiplateforme	Élevée (Windows, Linux, macOS).	Faible (environnement Office sous Windows uniquement).	Moyenne, nécessite recompilation selon la plate-forme.
Communauté et ressources	Très large, support actif, riche documentation.	Moyenne, centrée sur les usages bureautiques.	Large, mais plus technique et bas niveau.
Évolutivité et maintenabilité	Excellente (programmation modulaire, OOP).	Faible, difficile à maintenir dans de grands projets.	Moyenne à faible, gestion mémoire complexe.

TABLE 3.1 – Comparaison des langages de programmation : Python, VBA et C

À la lumière de cette analyse comparative, Python s'impose comme le choix le plus équilibré pour notre projet. Python est un langage haut niveau, interprété et multi-paradigme, ce qui en fait un excellent choix pour le développement rapide d'applications complexes. Il permet de combiner la programmation orientée objet avec des approches plus fonctionnelles, facilitant ainsi la structuration claire et modulaire du code.

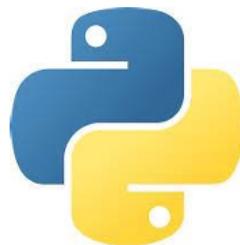


FIGURE 3.13 – Logo de Python

3.3.1.2 Environnement de développement VS Code

Le développement de l'application a été conçue en s'appuyant sur la distribution Anaconda, associée à l'environnement de développement intégré Visual Studio Code (VS Code). Grâce à son gestionnaire conda, Anaconda offre une gestion performante des bibliothèques et des dépendances, tout en permettant la création d'environnements virtuels isolés. Cela favorise la reproductibilité des projets et prévient les conflits de versions.

VS Code a été sélectionné en raison de sa légèreté, de sa souplesse et de ses multiples extensions destinées à Python, comme le débogage interactif, l'auto-complétion et l'intégration directe avec les environnements conda. Il propose également une prise en charge intégrée pour l'intégration de systèmes de gestion de versions tels que Git.

Cette sélection d'outils a garanti un environnement de développement structuré, durable et respectant les meilleures pratiques en matière d'ingénierie logicielle [9].



FIGURE 3.14 – Logo de Anaconda et VS Code

3.3.1.3 Framework PyQt5

Le framework PyQt5 a été employé pour créer l'UI de l'application. C'est une collection de liaisons Python pour la bibliothèque Qt5, considérée comme l'une des bibliothèques open source les plus performantes pour le développement d'interfaces utilisateurs multiplateformes.

Avec PyQt5, il est possible de créer des interfaces graphiques modernes, organisées et interactives tout en restant dans l'écosystème Python. Ce dernier propose une large sélection de widgets (boutons, zones de texte, menus, tableaux, etc.), tout en offrant l'opportunité d'ajuster précisément l'aspect et le fonctionnement de l'interface.

Son intégration aisée avec le langage Python permet de regrouper toute la logique de l'application et l'interface au sein d'un même espace, simplifiant ainsi la maintenance, les tests et le développement de l'application[10].



FIGURE 3.15 – Logo de PyQt

3.3.1.4 Personnalisation de l'UI avec QSS

Afin d'améliorer l'esthétique et l'ergonomie de l'interface graphique développée avec PyQt5, le langage de style **QSS (Qt Style Sheets)** a été utilisé. Inspiré de la syntaxe du CSS web, le QSS permet de modifier précisément l'apparence des composants de l'interface : couleurs, bordures, polices, effets au survol, coins arrondis, etc.

Cette approche permet de dissocier la logique fonctionnelle du design visuel, tout en maintenant un code propre et maintenable. Grâce à QSS, l'interface a pu être personnalisée pour correspondre à une identité visuelle cohérente et agréable à l'utilisateur, améliorant ainsi l'expérience globale [11].

3.3.1.5 Manipulation de fichiers Word avec python-docx

Dans le cadre du traitement automatisé des exigences, la bibliothèque `python-docx` a été utilisée pour manipuler des fichiers Word au format `.docx`. Cette bibliothèque permet de lire, parcourir et extraire le contenu des documents structurés, notamment les paragraphes, les tableaux, les cellules, et les styles associés.

Elle offre une interface haut niveau pour naviguer dans la hiérarchie du document, ce qui a été essentiel pour identifier les blocs pertinents contenant les "exigences". Grâce à `python-docx`, il a été possible d'implémenter un traitement automatique robuste capable de localiser des tableaux spécifiques, de filtrer les données non pertinentes, et d'extraire les expressions logiques nécessaires à l'analyse[12].

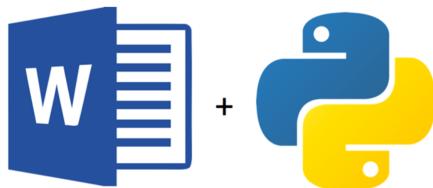


FIGURE 3.16 – python-docx

3.3.1.6 Manipulation de fichiers Excel avec pandas

La bibliothèque `pandas` a été utilisée pour la lecture et le traitement des fichiers Excel. Reconnue pour ses puissantes fonctionnalités de manipulation de données tabulaires, `pandas` permet d'importer facilement les feuilles de calcul sous forme de DataFrames, structures de données optimisées pour le filtrage, le tri, ou l'analyse d'informations[13].

Dans le contexte de ce projet, `pandas` a permis d'extraire automatiquement les informations pertinentes des "projets" contenues dans des tableaux Excel, de détecter les colonnes d'intérêt, d'effectuer des opérations de nettoyage, etc. La compatibilité native avec Excel via la fonction `read_excel()` a facilité l'intégration de ces fichiers dans le processus global de traitement, sans avoir recours à des outils externes.

Grâce à sa souplesse et à ses performances, `pandas` a joué un rôle essentiel dans l'automatisation du traitement des "projets".



FIGURE 3.17 – Logo de pandas

3.3.1.7 SQLite

Pour la gestion locale et structurée des données, le projet s'appuie sur le système de gestion de base de données SQLite. Il s'agit d'un SGBD (Système de Gestion de Base de Données) relationnel léger, embarqué, sans serveur, qui stocke l'ensemble de la base dans un simple fichier .db. Cette solution présente l'avantage d'être facile à intégrer dans une application Python, sans nécessiter d'installation supplémentaire ou de configuration complexe[14].

L'utilisation de SQLite permet de centraliser les informations traitées et d'effectuer des requêtes SQL (Structured Query Language) efficaces pour la consultation ou l'analyse des données.

L'interaction avec la base a été réalisée à l'aide du module standard `sqlite3` de Python, assurant une compatibilité native et une grande souplesse de manipulation.

Ce choix s'est révélé particulièrement pertinent dans le contexte du projet, notamment pour remédier aux problèmes de latence rencontrés lors de la manipulation de fichiers volumineux. L'adoption d'une solution légère, autonome et facilement déployable comme SQLite a ainsi permis d'optimiser les performances tout en assurant une gestion efficace des données.



FIGURE 3.18 – Logo de SQLite

3.3.1.8 Protocole SMTP

Le protocole SMTP (Simple Mail Transfer Protocol) est un protocole de communication standard utilisé pour le transfert de courriers électroniques[15].

Dans ce projet, il a été exploité via le module `smtplib` de Python pour envoyer automatiquement des emails de demande et de confirmation à l'administrateur lors de la suppression de projets. Ce mécanisme permet une interaction sécurisée avec un serveur de messagerie (par exemple Gmail), avec authentification et envoi structuré du message.

SMTP joue ainsi un rôle essentiel dans la gestion des notifications automatisées et garantit une traçabilité fiable dans le processus de validation.

3.3.1.9 PyInstaller

Afin de faciliter la distribution de l'application sans nécessiter d'environnement Python préinstallé, les scripts ont été convertis en exécutable autonome (.exe) à l'aide de l'outil PyInstaller.

PyInstaller est une bibliothèque Python qui analyse les dépendances d'un script, puis génère un dossier contenant un exécutable ou un fichier unique prêt à être lancé sur n'importe quelle machine Windows [16].

3.4 Réalisation de l'application

3.4.1 Menu Principal

Le menu principal constitue l'interface d'accueil de l'application et centralise l'accès à ses fonctionnalités clés. Il permet à l'utilisateur, via une navigation claire et intuitive, d'insérer de nouvelles exigences, de comparer logiquement celles déjà insérées, de consulter la base actuelle des exigences, de vérifier leur applicabilité par rapport aux

projets enregistrés, et enfin de gérer les fichiers de données liés aux exigences et aux projets. Voici un aperçu de menu principal :

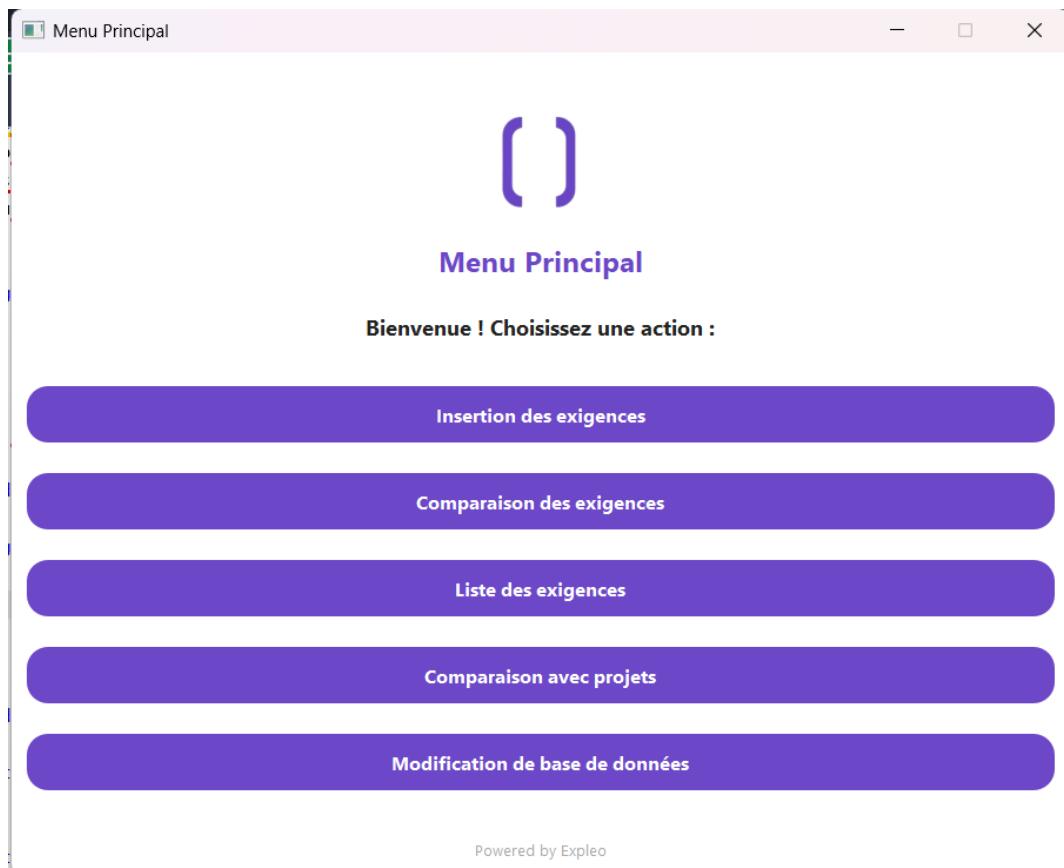


FIGURE 3.19 – Menu Principal de l'application

3.4.2 Interface d'insertion des exigences

Cette interface permet à l'utilisateur d'insérer une exigence spécifique en saisissant son ID unique au format standardisé. Une fois l'ID entré, l'application procède à l'extraction automatique des thématiques associées à l'exigence, en vue de sa comparaison avec d'autres. Deux boutons permettent soit de lancer la comparaison, soit de revenir au menu principal. Voici un aperçu de l'interface d'insertion des exigences :

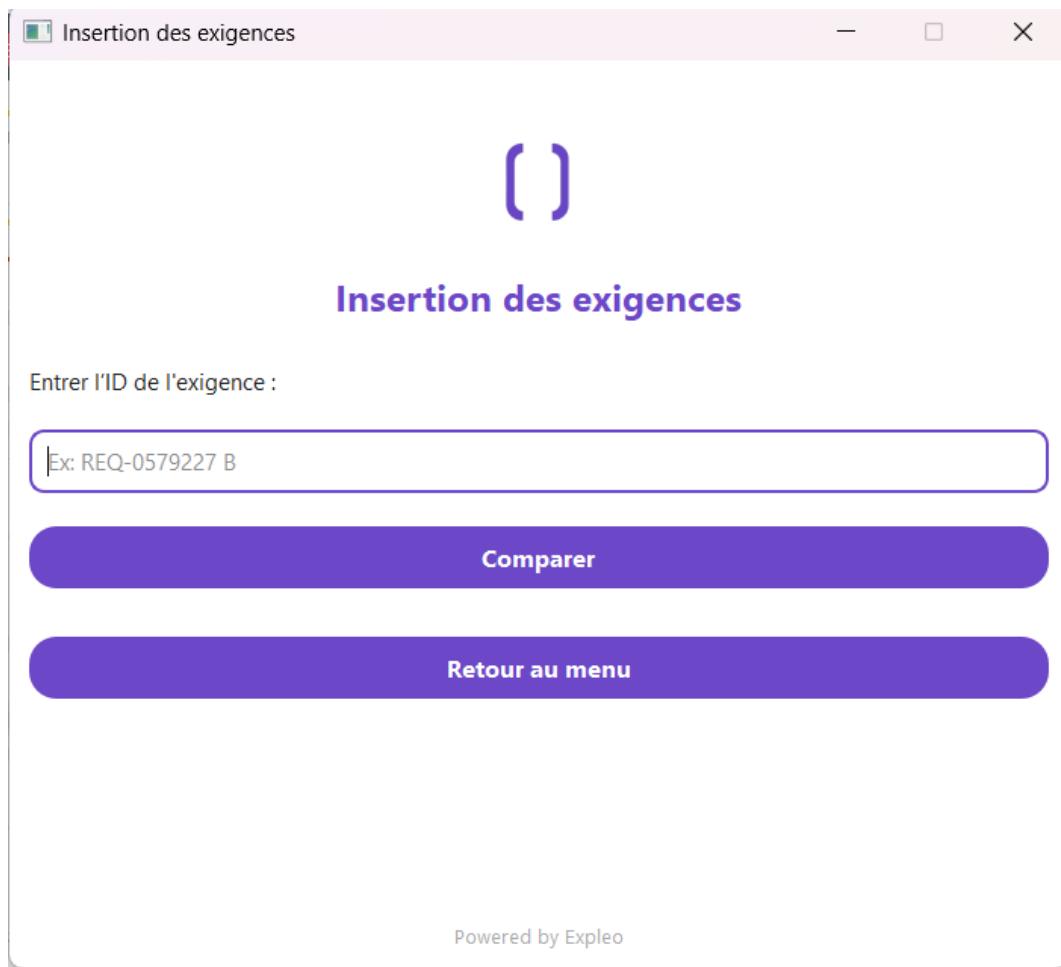


FIGURE 3.20 – Interface d’insertion des exigences

3.4.3 Interface de comparaison des exigences

Cette interface affiche le résultat de la comparaison logique entre les exigences insérées. Elle présente côté à côté le dernier résultat obtenu et le nouveau résultat, sous forme de combinaisons de thématiques extraites. L’utilisateur peut ainsi visualiser les évolutions entre les exigences comparées. Trois boutons sont disponibles pour revenir au menu principal, poursuivre l’insertion d’exigences ou consulter la liste complète des exigences. Voici un aperçu de l’interface de comparaison des exigences :

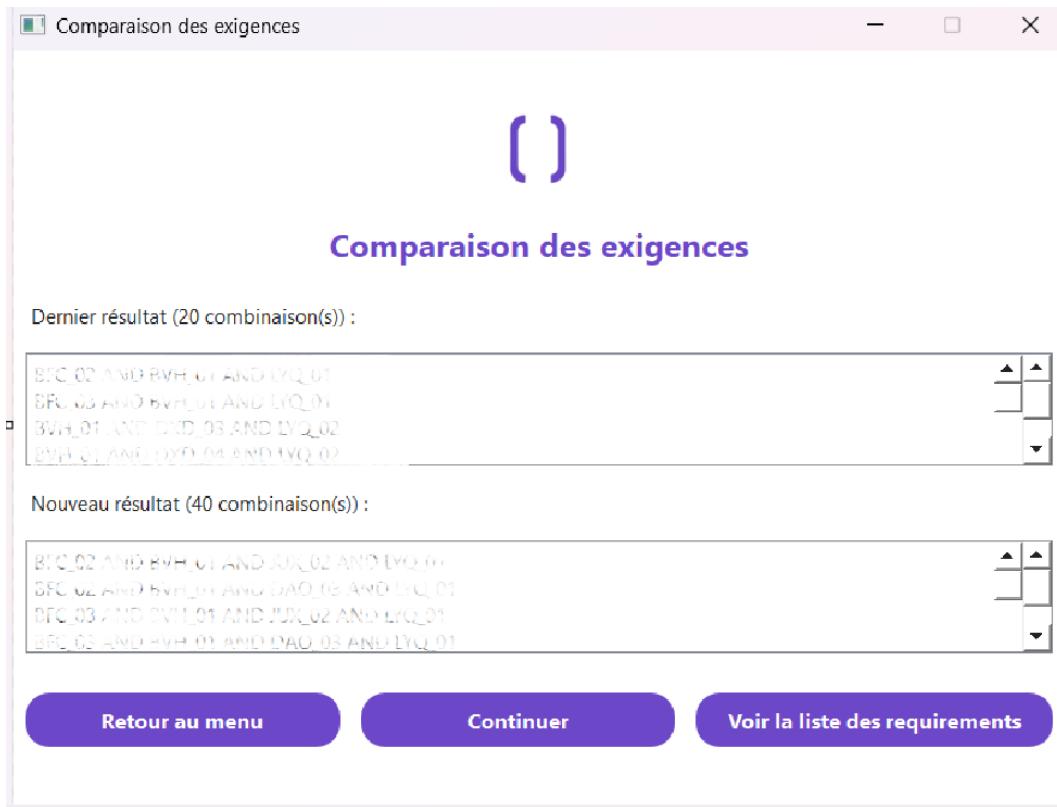


FIGURE 3.21 – Interface de comparaison des exigences

3.4.4 Interface de la liste des exigences

Cette interface affiche l'ensemble des exigences actuellement insérées dans la base de données. Chaque exigence y est représentée par son identifiant, sa date d'insertion, sa source d'origine (fichier), ainsi que les groupes de thématiques extraits. L'utilisateur peut consulter le contenu détaillé de chaque exigence, sélectionner une ou plusieurs entrées, puis les supprimer si nécessaire. Un bouton permet également de revenir au menu principal. Voici un aperçu de l'interface de la liste des exigences :

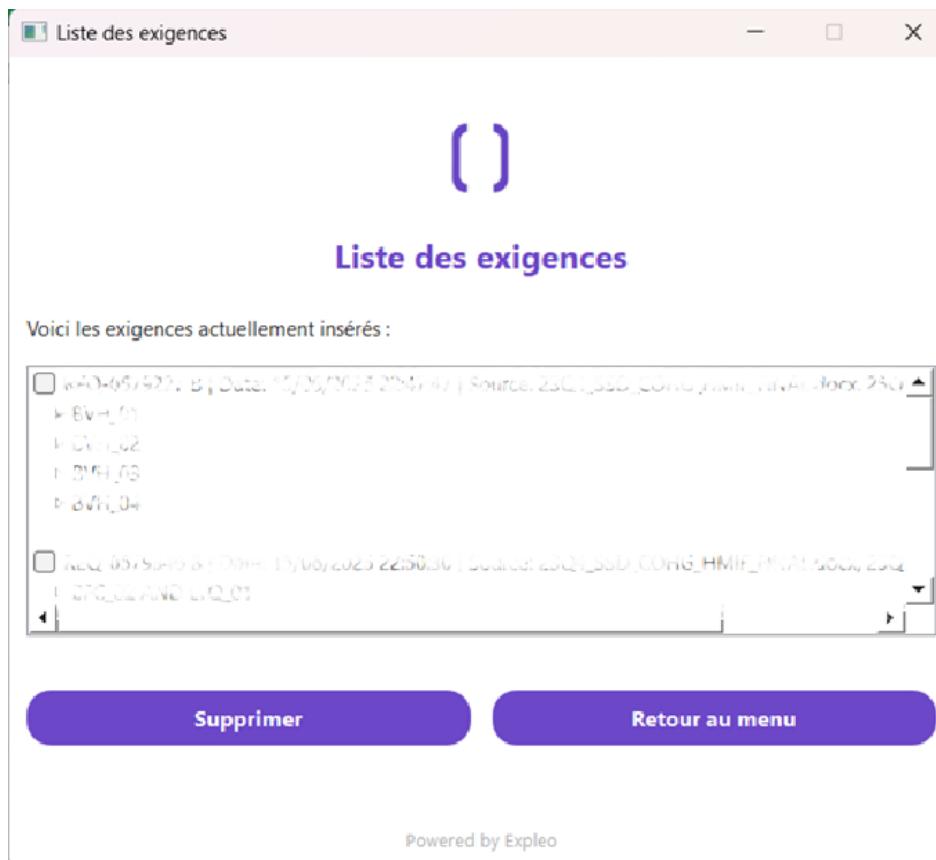


FIGURE 3.22 – Interface de la liste des exigences

3.4.5 Interface Comparaison avec projets

Résultat par projet : Cette interface permet de vérifier l'applicabilité des combinaisons d'exigences par projet. L'utilisateur peut rechercher un projet dans la liste, puis visualiser, pour ce projet, les combinaisons qui sont applicables et celles qui ne le sont pas. Cette analyse s'appuie sur les données de la base liée aux projets et aide à identifier les exigences potentiellement incompatibles.

Résumé global : Cette interface affiche un récapitulatif global des combinaisons d'exigences applicables à tous les projets présents dans la base.

Voici un aperçu de l'interface de résultat par projet et l'interface de résumé global :

3.4. RÉALISATION DE L'APPLICATION

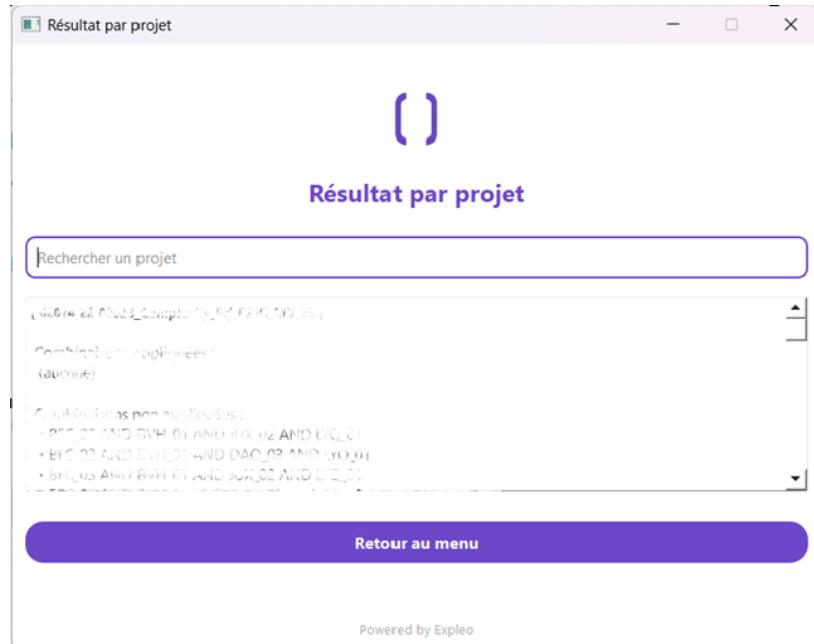


FIGURE 3.23 – Interface de Résultat par projet

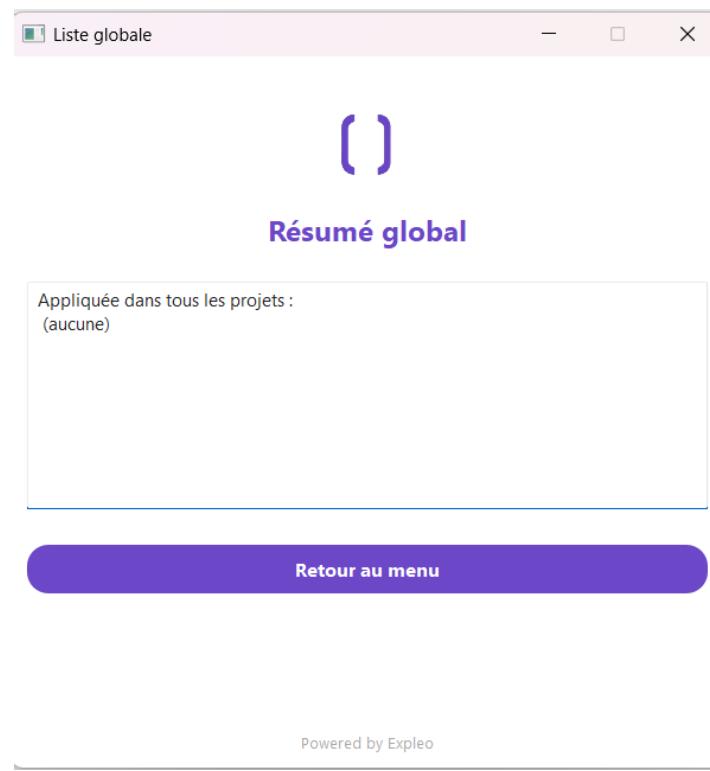


FIGURE 3.24 – Interface de Résumé global

3.4.6 Interface de Modification de la base de données

Cette interface permet de modifier le contenu de la base de données en accédant à deux types de gestion : d'une part, l'ajout d'exigences à partir de fichiers Word, et d'autre part, l'ajout ou la suppression de projets à partir de fichiers Excel, afin de maintenir la base à jour en fonction des besoins d'analyse. Voici un aperçu de l'interface de modification de la base de données :

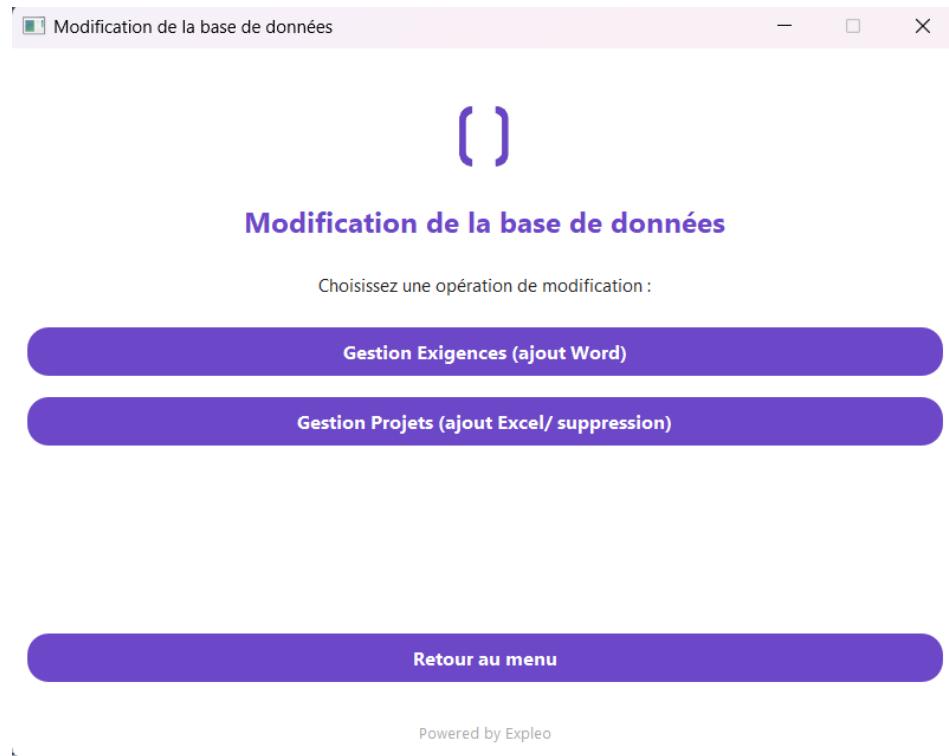


FIGURE 3.25 – Interface de Modification de la base de données

3.4.7 Interface de Gestion des Projets

Cette interface offre à l'utilisateur la possibilité de gérer les projets présents dans la base de données. Deux actions sont proposées : ajouter de nouveaux projets à partir de fichiers Excel contenant des informations d'applicabilité, ou supprimer des projets existants. Voici un aperçu de l'interface de gestion des projets :

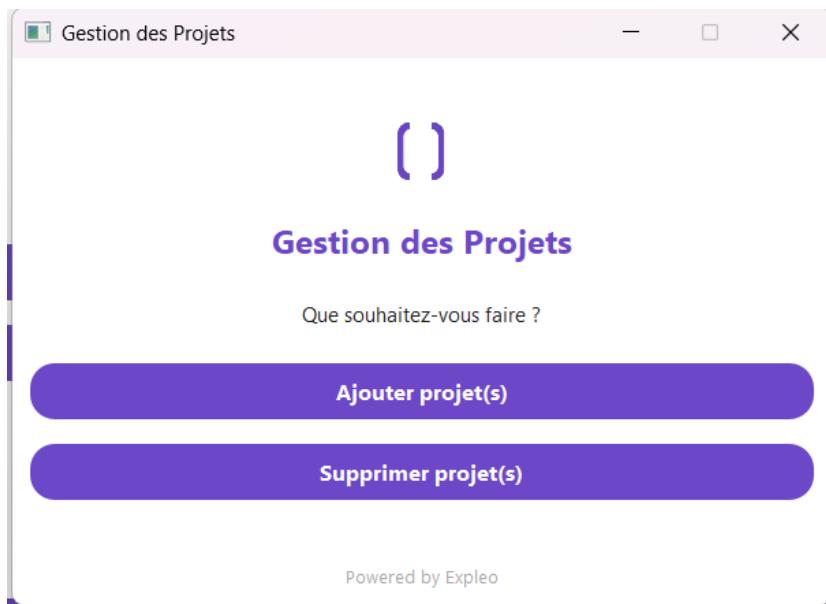


FIGURE 3.26 – Interface de Gestion des Projets

3.4.8 Interface de Suppression des projets

Cette interface permet de supprimer un ou plusieurs projets de la base de données. Après avoir sélectionné les projets à retirer, l'utilisateur doit obligatoirement confirmer son identité via un formulaire de validation comprenant nom, prénom et adresse email. Cette étape de confirmation renforce la traçabilité et la sécurité des opérations de suppression, en assurant un contrôle préalable avant toute modification irréversible des données. Voici un aperçu de l'interface de suppression des projets et l'interface de la demande de confirmation d'admin :

3.4. RÉALISATION DE L'APPLICATION

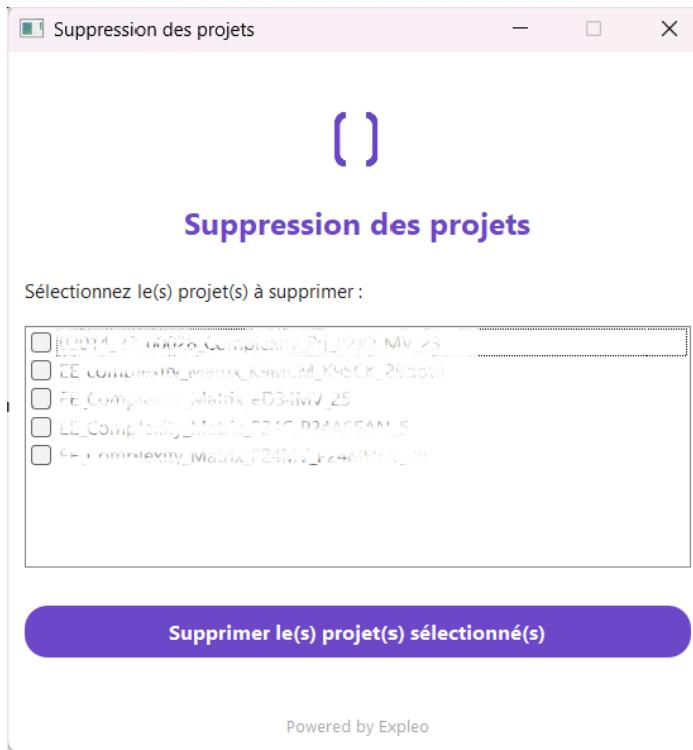


FIGURE 3.27 – Interface de Suppression des projets

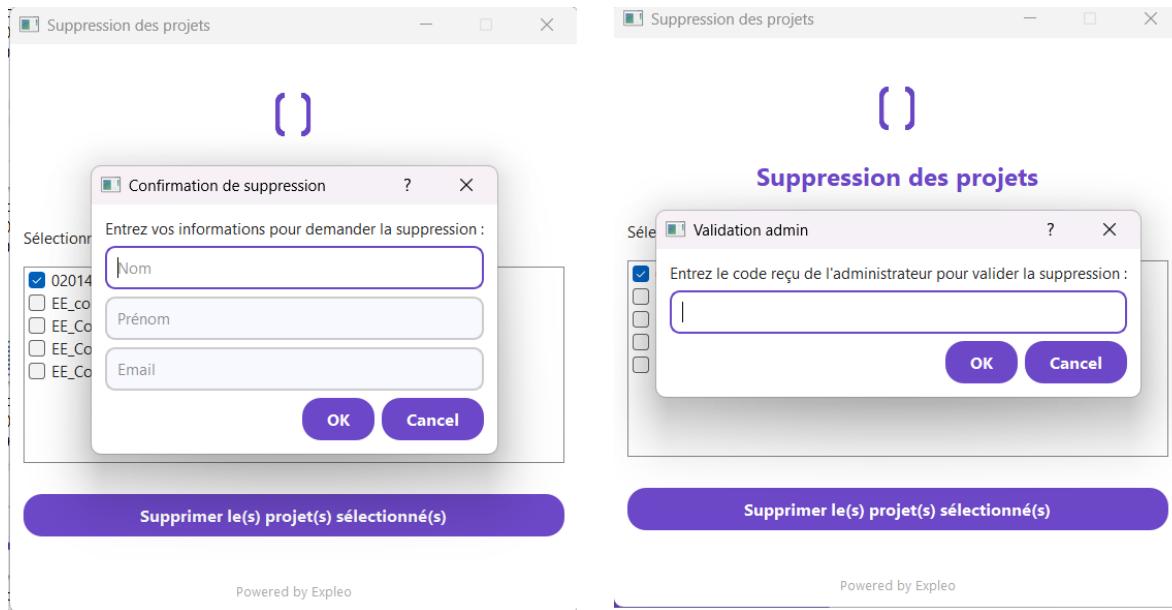


FIGURE 3.28 – Confirmation d'admin avant suppression

3.5 Conclusion

La conception et la réalisation de l'application ont été menées de manière méthodique, en s'appuyant sur une modélisation UML rigoureuse et une sélection d'outils adaptés aux besoins du projet. Cette approche a permis de structurer clairement le système, de maîtriser sa complexité, et de garantir sa maintenabilité.

Le choix du langage Python, combiné à PyQt5 pour l'interface graphique et SQLite pour la gestion des données, a facilité le développement rapide d'une application performante et multiplateforme. L'intégration de bibliothèques spécialisées comme python-docx ou pandas a également permis une automatisation fiable des traitements de documents techniques.

Au terme de cette phase, l'application offre une interface complète permettant à l'utilisateur de gérer efficacement les exigences, de les comparer selon des règles logiques, de vérifier leur applicabilité aux projets, et de suivre l'évolution des données grâce à une traçabilité renforcée. Ce support technique robuste constitue une base solide pour de futures extensions ou déploiements industriels.

CONCLUSION GÉNÉRALE

Au terme de ce projet, nous avons pu répondre de manière concrète à une problématique réelle rencontrée dans les processus de validation des calculateurs automobiles, en particulier le BSI, chez Expleo Group. En concevant et développant une application logicielle sur mesure, nous avons permis d'automatiser une tâche critique : la vérification de l'applicabilité des exigences fonctionnelles à différents projets véhicules.

Grâce à une analyse approfondie du contexte industriel, une modélisation UML structurée, et l'adoption d'outils performants comme Python, PyQt5 et SQLite, l'application développée est à la fois intuitive, efficace et conforme aux attentes. Elle intègre des fonctionnalités variées allant de l'extraction des exigences depuis des documents complexes, jusqu'à leur comparaison logique et leur validation croisée avec des projets multiples.

La méthode d'extraction mise en œuvre, fondée sur une logique déterministe, a été choisie en raison de la structure complexe et spécifique des documents manipulés. Ce choix s'est révélé particulièrement robuste et adapté au contexte actuel. Néanmoins, afin d'élargir le champ d'utilisation de l'application, une évolution pertinente consisterait à intégrer des techniques d'intelligence artificielle, notamment le NLP (Natural Language Processing). Cela permettrait de généraliser l'analyse à d'autres types de documents, qu'ils soient non structurés, textuels ou hétérogènes, et d'automatiser la détection d'exigences dans des contextes où une approche purement structurelle atteint ses limites. Une telle piste d'amélioration ouvrirait la voie à une application plus flexible, capable de s'adapter à la diversité des formats documentaires et aux exigences d'industrialisation à grande échelle.

BIBLIOGRAPHIE

- [1] *expleo*. URL : <https://expleo.com/global/fr/>.
- [2] *Rapport de stage : amélioration et optimisation des processus de validation du calculateur BSI*, Réalisé au sein de Expleo Group, par Ilyas MADANE.
- [3] *Voiture hybride : quelle différence entre HEV, MHEV, PHEV ?* URL : <https://www.actu-automobile.com/2022/08/10/voiture-hybridequelle-difference-entre-hev-mhev-phev/>.
- [4] *Fuel Cell Electric Vehicles*. URL : <https://afdc.energy.gov/vehicles/fuel-cell>.
- [5] *Vector*. URL : <https://www.vector.com/int/en/know-how/communication/>.
- [6] *Diagnostics over IP*. URL : <https://media.lidcdn.com/dms/document/media/v2/D4D1FAQFpkheFX1RNJg/feedshare-document-pdf-analyzed/B4DZc1vL93HAAc-0/1748953325038?e=1750291200&v=beta&t=j-03lj7DvHeNAu04JiiJHR7tyc2-Sw1KR9gx1Q31VuU>.
- [7] *Architecture Phases Vie et Reseaux du BSI*. URL : https://wiki.inetpsa.com/ASRP/Home/SWactivities/Architecture#PhasesVie_et_Reseaux_du_BSI.
- [8] *step-lab*). URL : <https://step-lab.com/fr/quest-ce-que-le-hil/>.
- [9] *anaconda*. URL : <https://www.anaconda.com/docs/main>.
- [10] *PyQt5*. URL : <https://www.riverbankcomputing.com/static/Docs/PyQt5/>.
- [11] *QSS*. URL : <https://doc.qt.io/archives/qt-5.15/stylesheets-reference.html>.
- [12] *docx*. URL : <https://python-docx.readthedocs.io/en/latest/>.

- [13] *pandas*. URL : <https://pandas.pydata.org/>.
- [14] *sqlite*. URL : <https://www.sqlite.org/docs.html>.
- [15] *SMTP*. URL : <https://docs.python.org/3/library/smtplib.html>.
- [16] *PyInstaller*. URL : <https://pyinstaller.org/en/stable/>.