RAPPORT DE STAGE

Sommaire :

I / Introduction : 3

II / Solutions proposées : 4

III / Problèmes rencontrés : 5

IV / Architecture globale : 6

V / Algorithmes : 6

VI / Compte rendu des rendez-vous : 7

VII / Conclusion : 8

VIII/ Glossaire : 9

IX / Références : 10

Introduction:

Un diagnostic embarqué, en anglais On-Board Diagnostics (abrégé en OBD), est un système embarqué dans la plupart des véhicules produits à partir des années 2000.

Il permet à un acteur externe d'avoir accès aux informations collectées par l'ensemble des composants du groupe motopropulseur.

Dans le cadre des projets tutorés du département informatique universitaire de technologie de Villetaneuse, nous avons choisi de développer un logiciel de diagnostic OBD-II pour proposer à un utilisateur quelconque une vision lisible et clair des informations de sa voiture.

Nous avons choisi ce sujet car nous avons estimé qu'il pouvait nous présenter des problématiques nouvelles et intéressantes, notamment la partie communication à travers flux que peu d'entre nous avaient eu l'occasion d'aborder lors de la première année, ainsi qu'une première approche à l'intelligence artificielle via le développement possible d'un système expert.

Dans un premier lieu cependant, l'objectif était de permettre à l'utilisateur de réaliser du monitoring basique à travers une interface concise, intuitive et compatible sur tout système d'exploitation.

Solutions :

Compte tenu de notre statut d'étudiant et étant conscients que nos compétences allaient nécessairement évoluer au cours de ce projet, nous avons choisi d'adopter la méthode SCRUM pour sa flexibilité.

Versioning :

Notre premier contact avec le versioning s'étant réalisé par une introduction à Github, dont nous détenions la plupart des outils au moment de commencer le projet, nous nous y sommes donc naturellement dirigés pour partager notre code.

Etablissement des tâches:

Nous avons découpé les tâches de la manière suivante :

\*

tâches

\*

Ce découpement n'a nécessité que peu de recherches, ce qui nous a permis de rapidement nous répartir le travail et de passer à la première phase de développement.

Problèmes rencontrés :

Notre premier problème a été d'établir une connexion bluetooth entre le périphérique de l'utilisateur et le module ELM327. En effet, après un prise en main prudente de l'API bluetooth proposée par Java l'identification du protocole contenant le service auquel se connecter s'est fait un peu à taton. Sans parler du type de connexion à établir, qui nous a demandé quelques recherches avant de comprendre que nous avions besoin d'une connexion client.

L'architecture de "l'engine" soccupant de gérer l'envoi et la lecture des requêtes nous a également longtemps laissé perplexe. Le mode 01 comprenant de nombreux PIDs, et le module répondant spécifiquement à quasiment chacunes d'entres elles (parfois avec un float encodé sur un Byte, d'autres avec plusieurs String encodés sur quatre Bytes etc...), notre première approche a donc été de créer une classe par PID de requête.

\*UML\*

Cette méthode s'est cependant rapidement révélée extrêmement lourde en terme de développement et incroyablement rigide, aussi avons nous fini après plusieurs expérimentations à développer une simili table de routage constante réalisant la liaison entre le nom d'une information, son numéro de PID et le traitement associé à la réponse de son envoi. Certains d'entre nous ayant déjà commencé à avancer sur le développement de l'interface graphique, nous avons alors largement profité de la souplesse dont bénéficiait la conception du projet, ce changement d'architecture n'opérant aucune modification nécessaire à ce niveau du code.

\*UML\*

Au niveau du traitement là encore, nous avons souhaité rendre aussi facile que possible l'implémentation d'un nouveau PID à la table, sachant que cela nous sauverait beaucoup de temps. Nous avons donc développé un automate traduisant une chaîne de caractère en opération arithmétique et retournant le résultat.

\*Code\*

Un autre problème rencontré a été le remplissage du buffer en entrée, car en lui donnant tout ce que le module nous retournait, nous nous retrouvions avec de nombreuses informations que nous n'avions pas d'intérêt à exploiter. Nous avons donc mis en oeuvre quelques filtres au niveau de la lecture, ne remplissant ainsi le buffer qu'avec les Bytes encodant la donnée désirée.

Architecture globale du code :

Nous nous sommes rapidement intéressé au design pattern que nous commencions à voir au début du projet, et avons ainsi tenté de reproduire un pattern Modèle Vue Contrôleur, pour pouvoir se répartir sur l'ensemble du projet dès le début.

\*Screen eclipse\*

\*UML\*

Algorithmes:

Compte rendu des rendez-vous:

premier rendez-vous :

Principalement la phase de conception : nous avons présenté la répartition des tâches sur le temps ainsi qu'une estimation de leur coût à travers un premier diagramme de Gantt.

\*Documents\*

second rendez-vous:

Nous avons réussi à établir une connexion entre le

logiciel et le module. Une première ébauche de l'architecture de la gestion des requêtes a également été présentée, ainsi qu'un diagramme de classe pour le code déjà créé.

\*Documents\*

troisième rendez-vous:

Nous avons pu faire une démonstration en bêta de l'affichage graphique final, n'offrant que la possibilité d'observer la courbe de vitesse de son véhicule.

\*Documents\*

Conclusion:

Ce projet nous a offert la possibilité de découvrir plus profondément les fonctionnalités et les enjeux de la programmation orientée objet en mettant en pratique nos connaissances fraîchement acquises en conception, ce qui s'est avéré extrêmement intéressant. La prise en main de l'API bluetooth offerte par Java nous a également entraîné à composer à partir d'un code inconnu en lisant la documentation.

Nous sommes cependant bien conscient que de nombreuses fonctionnalités pourraient et devraient être ajoutées à notre produit afin de le rendre véritablement utile à l'utilisateur, comme la possibilité de mettre en rapport plusieurs informations sur un seul graphe, ou le développement d'un véritable système expert pour proposer à l'utilisateur où pourrait se trouver l'erreur au lieu de simplement lui afficher le code erreur.

Annexe:

Glossaires:

Références: