Projet dilemme du prisonnier

Rapport

CPE Lyon

3ème année de Formation Ingénieur Informatique et CyberSécurité

Clément COURIOL
Cédric GABETTE
Louis MORAND
Mathis FRANCFORT

Sommaire

1 - Introduction

Ce rapport présente les principales informations relatives aux méthodes de travail et de développement utilisées lors du projet d'implémentation du dilemme du prisonnier. Ce projet a été réalisé en groupe de quatre personnes, basé sur un cahier des charges fourni spécifiant les fonctionnalités attendues.

2 - Organisation du travail dans le groupe

Niveau organisation, deux groupes ont été formés pour développer en parallèle l'application client et serveur, ceci permettant d'avoir un avancement constant dans le projet et d'optimiser notre temps de travail : un groupe client (Cédric, Clément) et un groupe serveur (Louis, Mathys).

Pour la prise en main du projet nous avons d'abord discuté et réfléchi sur la manière dont on allait réaliser le projet, sur le fonctionnement de notre programme, afin de pouvoir se donner une ligne directrice et définir clairement nos objectifs. Nous avons mis en place des diagrammes, des schémas sur un tableau, identifié les différents problèmes que l'on pouvait rencontrer. C'est une fois cette étape réalisée que nous avons commencé à écrire notre programme.

Deux branches ont ensuite été créées sur git, une branche client et une branche serveur sur lesquelles travaillaient chacun des deux groupes. Le but étant que chaque groupe puisse travailler sans modifier involontairement le travail de l'autre ainsi que d'apporter des modifications régulières. Le fait de toujours rester en contact et d'informer les autres de chaque modification permet d'éviter des problèmes liés entre autres aux structures. Cela a été réalisé à travers la mise en place d'un serveur discord qui nous a servi pour la discussion sur le projet et les différents rendez-vous à se donner pour la mise au point sur l'avancement du projet.

De nouvelles branches de test ont été créées par la suite pour fusionner le travail des deux groupes et vérifier le bon fonctionnement du programme.

Pour la gestion des fonctionnalités à réaliser, nous avons choisi de noter cela dans le README du projet en ajoutant et cochant les différentes tâches effectuées ou a effectuées.

3 - Méthodologie

Pour le fonctionnement du programme, nous avons dans un premier temps réalisé différents diagrammes UML pour nous aider dans le développement du projet ainsi que pour prévoir les différents problèmes. L'utilisation de GitHub nous permettait de suivre à distance les différentes modifications apportées au programme du projet. (voir annexe 1 et 2). Le diagramme UML du protocole d'échange, en annexe 2, est une version plus détaillée qui reflète les différentes étapes de jeu telles qu'elles ont été implémentées.

Au début de chaque séance, les deux membres de chaque groupe se mettaient d'accord sur des tâches à effectuer, ce qui peut donc s'apparenter à des sprints d'une durée de 4 heures. Ces tâches peuvent être de tout genre, telles que la conception, le débogage, ou la documentation, entre autres. Cette méthode de travail permet une certaine autonomie. Si problème il y a, une harmonisation ou une modification est effectuée (dans le cas par exemple d'un appel de fonction n'ayant pas le bon nom ou d'un ajout d'option dans une fonction).

4 - Justification des choix et la pertinence à posteriori

- Fichier de configuration au lieu d'une entrée client pour la configuration de l'ip et du port via GUI: On utilise un fichier de configuration pour automatiser la connexion des clients au serveur et pour définir ses paramètres, ce qui permet de limiter les erreurs d'exécution du client.

- Passer par des structures pour les données au lieu de JSON / XML pour l'envoi des données: Le choix d'utiliser des structures au lieu d'autres formats de données, est utilisé pour des raisons de simplicité sur l'implémentation, et de rapidité au niveau de l'encodage, cependant il y a des possibilités de perte de portabilité à cause de la taille des types utilisés selon les machines, pour pallier à ce problème une sérialisation des données pourrait être mise en place. Cependant cette solution n'a pas été mise en place à cause de la difficulté d'implémentation.

- La façon de stocker les clients sur le serveur: Le stockage des clients dans un tableau spécifique permet de facilement réaliser des opérations sur ces derniers, cependant la place prise en mémoire est importante, ce qui est problématique à long terme.

- La communication inter thread (en ce moment booléens, pourquoi pas sémaphores ou pipes): La communication inter thread est complexe, lors du début du projet nous avions décidé de faire au plus simple et d'éviter les sujets trop mal maîtrisés, tel que les pipes ou mémoire mappée, l'objectif étant de rendre un projet fonctionnel avec possibilité d'évolution. Le choix de variables globales est une idée exécutable sur de petits programmes, comme ici, mais dès que ce dernier gagnera en complexité, d'autres types de communication plus pratiques devront être mises en place, avec par exemple des sémaphores, qui pourraient convenir pour les données envoyées.

5 - Statut du projet

Actuellement, le projet est fonctionnel en répondant aux points majeurs à respecter, qui sont la communication entre les clients, et la sauvegarde sur un fichier. Après avoir lancé le serveur, ce dernier se met à l'écoute du client. Une fois que deux clients sont lancés, le serveur les met en relation pour la partie et échanges les différentes données. Enfin une fois la partie finie, les clients se ferment, un log de la partie est écrit dans un fichier.
Cependant, plusieurs améliorations peuvent être apportées au projet:

- Afficher au client s'il a gagné ou perdu le round
- Afficher au client d'une recherche de partie quand il n'y a pas de pair disponible
- Gestion de plusieurs paires de clients
- Sérialisation pour les échanges client/serveur
- Proposer une nouvelle partie aux clients
- Afficher la valeur de la somme du client dans la gui
- Ajouter une possibilité de mise manuelle pour le client
- Ajouter le support de configuration via cli au lieu d'un fichier
- Ajouter le support de la modification du fichier de configuration par cli
- Typer correctement les variables
- Nommages normé des variables / structures entre le client et le serveur

6 - Difficulté rencontrées

Côté serveur :

Les difficultés rencontrées ont été le débogage de certaines fonctions suite à des messages d'erreur qui nous étaient inconnus jusqu'à lors. Étant habitués à ce que chacun travaille sur sa propre branche, le fait de travailler sur la même branche qu'un collègue nous a forcé à devoir gérer les merge conflict des push-pull. Nous avons également dû revoir la notion de thread. Ce projet étant le premier réalisé avec autant de personnes (le double du nombre habituel), il a donc nécessité un bon travail d'organisation, de communication, et de partage du travail au sein du groupe.

Côté client :

De manière classique nous avons réalisé du débogage sur certaines fonctions et fonctionnalités. Il y avait 2 aspects importants pour le développement du client: la mise en place d'une GUI, ainsi que le programme qui communiquera avec le serveur. Sur ce point, la difficulté était de pouvoir gérer la communication du programme avec la GUI en simultané avec le serveur pour transmettre les données au travers des threads. Nous avions aussi pensé que le client réaliserait un minimum de traitement mais au final, celà rendait la programmation inutilement compliquée et nous avons décidé de transférer ces fonctionnalités au serveur et ceci paraît plus adéquat au système client/serveur.

Pour la gestion des différentes tâches à faire, il aurait été plus judicieux d'utiliser un outil spécialisé comme Trello, Notion, Clickup ou bien la gestion de projet intégré directement dans github. Lié les tâches effectuées au README est une mauvaise idée car la mise à jour nécessite de créer un commit qui n'est pas appliqué à toutes les branches,

rendant le suivi moins efficace. Cela crée des commits, push et merge inutiles qui n'ont pas à être dans le version control du code avec git.

7 - Bilan

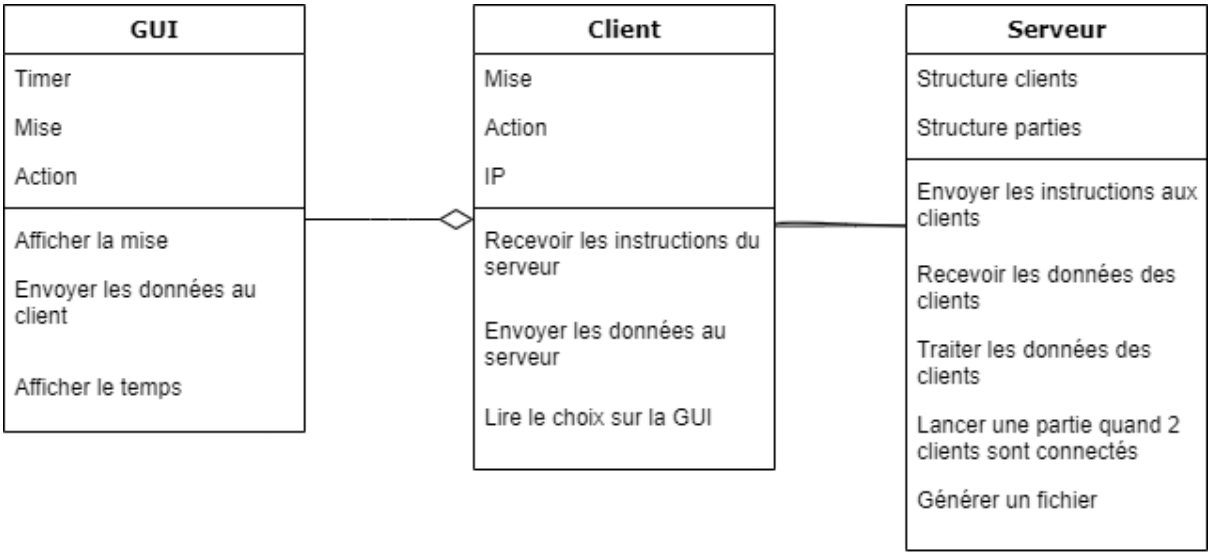
Ce projet nous a apporté à tous une grande expérience dans le travail en groupe, avec des contraintes de temps et d'organisation, que ce soit à travers la gestion d'un projet avec git, qu'avec des rendez vous à se donner pour faire le point sur l'avancement.

Il nous a également permis de comprendre le fonctionnement général d'une application client/serveur avec un serveur et plusieurs clients, tant sur le point du fonctionnement général, à travers la gestion des clients, de leurs connexions, et des rôles et fonctionnalités du serveur, que des différentes problématiques que ce type d'architecture présente.

Cela nous a permis de gagner en expérience, que ce soit au niveau des capacités de programmation, avec la programmation du client et du serveur, ou bien de l'interface graphique avec GTK, que de la gestion de travail communautaire via l'outil Git.

Annexes

Annexe 1 : Diagramme de classe du programme



Annexe 2 : Seconde version du schéma de communication client/serveur