Bilan de gestion d'équipe et de projet

Introduction/ouverture:

Il est maintenant temps de faire un retour rétrospectif sur le travail effectué ainsi que sur l'organisation d'équipe.
Il faut considérer plusieurs aspects du projet, celui relevant de l'humain et celui lié à la technique, à la conception *stricto sensu* du compilateur.
Le travail de groupe c'est avant tout des échanges, des paroles, des discussions. Nous avons fait le choix de venir tous les jours en présentiel afin de bénéficier de cette richesse des interactions. Comme Aristote l'avait déjà observé dans *La Politique,* le travail crée de la richesse pour l'ensemble de la collectivité et, en ce sens, la renforce. Chacun est libre d'exprimer ses idées au sein du groupe, de les valoriser sans prétendre détenir la vérité absolue. Il n'y a pas lieu de faire prestance d'autorité puisque celui qui se dit détenir l'*Aletheia* au nom de la philosophie grecque se verra bien moqué. Soit.
Afin de développer le compilateur sereinement, il fallait bien entendu organiser et répartir le travail. Ce dernier est plus efficace lorsqu'il est divisé. Comme Adam Smith l'a illustré de son temps avec l'exemple d'une manufacture d'épingles, on est beaucoup plus efficace lorsqu'on travaille à plusieurs et que chacun se spécialise dans une tâche. Même si le temps du travail à la chaîne en Europe est presque révolu, ce concept est parfaitement d'actualité et pertinent aujourd'hui.

Considérer les objectifs:

Le meilleur moyen de se perdre ou de tourner en boucle dans un projet est lorsque les objectifs à atteindre sont flous ou mal fixés par les membres de l'équipe. Afin de se prémunir contre cette malencontreuse situation, il est préférable avant même de commencer à écrire la moindre ligne de code de fixer les directions où nous souhaitons aller, données en réalité par les attendus du projet. Comme l'illustre si bien le concept de la Guillotine de Hume, les méthodes ainsi que les moyens d'action utilisés ne seront discutés uniquement après le choix des objectifs qui eux ne sont pas objectifs, du moins les objectifs intermédiaires. Bien heureusement, la division du projet en étapes A, B

et C permet une segmentation intermédiaire permettant aux membres de l'équipe de se projeter à moyen terme.
De manière générale, il faut noter que les individus de l'équipe ne partent pas sur un pied d'égalité en termes de technique ou de compréhension des concepts de théorie des langages. Il existe alors des difficultés pour certains qui sont facultés pour d'autres, d'où la nécessité des échanges pour que tout le monde puisse évoluer vers le haut.

Organisation effective:

Il est tout d'abord notable d'indiquer que nous travaillons avec Git, en présentiel, avec différentes branches. *Master* est réservé au contenu destiné à être livré à l'évaluation alors que d'autres branches sont utilisées lors du processus de développement pour mettre en commun le code. L'utilisation de CodeWithMe de IntelliJ nous permet de limiter les requêtes Git susceptibles d'engendrer des merges parfois difficilement résolubles.

Concernant les documents dactylographiés, ces derniers étaient accessibles par l'ensemble des membres grâce à un lien Google Drive disponible sur notre serveur Discord. Ils étaient essentiellement constitués des différentes documentation à rédiger, des fichiers relevant les erreurs de grammaire et des idées de test à implémenter.

Il a été décidé comme suit concernant la répartition des tâches au début du projet, avec bien sur des flexibilités possibles s'il y a un besoin d'aide entre les équipes:

- Tout le monde travaille sur l'étape A:
  - Deux personnes élaborent le *Lexer*, c'est-à-dire complètent le fichier *DecaLexer.g4* contenu dans le dossier ***antlr4*** pendant que les 3 autres membres complètent le *Parser,* c'est-à-dire complètent le fichier *DecaParser.g4* contenu dans le même dossier.

Il se trouve que l'étape s'est déroulée sans grand encombre et sans prendre trop de temps, moins d'une semaine. Les tests de la catégorie

syntax, beaucoup moins nombreux que ceux de l'étape B, sont alors validés.
En effet, les parties les plus chronophages s'avèrent être les étapes B et C.
Le développement s'organise d'abord en considérant la partie sans objets (sans classe) puis une fois que cela est fonctionnel, la partie avec objets peut être abordée.
Il a été choisi de mettre trois personnes sur l'étape C pendant que les deux autres s'occupent de l'étape B. Le choix de mener deux batailles de front a été fait à des raisons de parallélisation du travail dans la mesure de gagner un peu de temps. Si l'un des membres a des questions sur une partie sur laquelle il n'a pas travaillé, il sera le bienvenu à la poser à l'autre groupe. Dans l'autre sens, il sera en mesure de répondre à ses camarades dans le questionnement puisqu'il se sera spécialisé dans sa partie du travail.

Concernant les personnes travaillant sur l'étape B, se sont dessinés deux profils avec le temps en fonction de nos appétences informatiques. Une personne travaillait plutôt sur le code Java du compilateur, les fonctions verify, decompile, c'est-à-dire la vérification contextuelle. L'autre personne s'occupait de réaliser et d'organiser l'architecture des tests *.deca*. Ainsi, la personne réalisant les tests permettait de relever les erreurs ou insuffisances du code Java réalisé par l'autre et de l'aider ensuite à en faire la correction. Ainsi, l'exécution de certains tests .deca individuellement permet d'affiner progressivement le code Java et la vérification contextuelle. Une fois l'étape B presque au point, un des membres s'est engagé à commencer la documentation pendant que l'autre poursuivait vers l'étape de debug en confrérie avec les personnes travaillant sur l'étape C.

Dans l'étape C, un étudiant s'est concentré sur le code du compilateur (une bonne partie ayant été faite durant les vacances), tandis que les deux autres en élaboraient les tests.
Lors de la conception des étapes avec objet et essentiel, l'un des étudiants s'étant concentré sur les tests s'est consacré essentiellement à l'étape A.

S'ensuit, vers la fin du projet (fin dernière semaine), une étape de debug et de rajout de tests commandités par un des membres qui avait besoin de tests supplémentaires concernant l'étape C qu'il était en train d'affiner.

Il est légitime de noter que le développement de ce compilateur ne s'est pas déroulé *stricto sensu* de manière purement linéaire suivant les étapes: analyse, conception, codage, validation, documentation, tel que nous aurions pu penser à le faire à la genèse du projet. Bien entendu il y a eu une analyse au début notamment concernant les erreurs possibles dans les règles de grammaire relevées sur feuille de note (google doc sur notre drive). La suite s'est déroulée plutôt comme une alternance entre validation des tests ajoutés et correction des erreurs ou insuffisances présentes dans le code Java du compilateur deca. De cette façon, il était possible de connaître les points validés et ceux à corriger. Il était aussi important de s'assurer de la non contingence des tests, les tests validés doivent rester opérationnels après amélioration du code, bien entendu. Cette pratique était souvent effectuée par la personne ayant pour tâche d'exécuter les tests.

<u>L'extension: ByteCode Java:</u>

Nouvel objet, nouveaux soucis, il a bien fallu se renseigner sur la place qu'occupe le ByteCode Java au sein du projet et plus généralement au sein de Java. Nous nous sommes penchés dessus au retour des vacances de Noël, pleins d'inspiration et de nouvelles idées innovantes.
Certains ont effectué des recherches internet et vidéos, et pris des notes sur feuilles afin de comprendre ce que représente le ByteCode Java, l'explication affinée est donnée dans la documentation sur l'extension.
A priori, la réalisation d'une telle extension nous paraissait plutôt laborieuse voire obscure. Il se trouve qu'un membre de l'équipe a eu, avec sagacité, l'ingénieuse idée d'utiliser le maximum d'outils existants et développés durant le projet, à savoir les fonctions *decompile* et les outils de génération de ByteCode de *javac*. Ceci a alors redonné espoir

et énergie à l'équipe dont les idées se sont remises à mûrir concernant l'extension. Elle a ainsi pu être complétée *in tempore*.

Documentation

Concernant la documentation, il était préférable que les personnes ayant travaillé de manière plus approfondie sur un aspect du projet rédige la partie de la documentation concernée. Par exemple, concernant les commandes utilisateurs, la personne ayant travaillé dessus *in corpore* sera plus à même d' en faire la description et les explications concernant leur utilisation.
Ainsi, la documentation s'est enrichie petit à petit et quiconque dans l'équipe était libre d'écrire des lignes supplémentaires concernant les concepts plus généraux tels que la gestion de projet et d'équipe qui ne nécessitent pas une connaissance technique approfondie.
De plus, la présence, oserions-nous dire "physique", de tous les membres dans la salle de travail permettait au rédacteur de la documentation de s'assurer auprès de ses camarades si ce qu'il pouvait écrire concernant leurs parties faisait bien sens. En effet, jusqu'au rendu du projet tout le monde n'était pas en train d'écrire la documentation en même temps mais en parallèle de rajout de tests par exemple.
C'est ainsi que du néant naquit la littérature.

Pour conclure sur ce bilan *a posteriori*, il est légitime de relever que l'aventure, malgré un calme apparent, a été plus houleuse pour certains que pour d'autres. Cependant, l'esprit dominant a été celui de l'entraide, de la camaraderie, de la fraternité puisque comme l'a si bien évoqué un grand philosophe "Seul on va plus vite, ensemble on va plus loin".

FIN