Bilan de gestion d'équipe et de projet

Equipe 10

January 2021

Contents

1 Introduction

Dans ce document, on va vous présenter une vue d'ensemble sur le déroulement du projet pendant les trois semaines déjà passées. Notre équipe est composée de cinq membres venant de différentes filières, la plupart se connaissant, ce qui a garanti une bonne ambiance au sein de l'équipe et a certainement facilité la phase de démarrage du projet. Dès le début, nous avons essayé de repérer les forces et les faiblesses de chacun afin de devenir efficaces aussi vite que possible. Pour mieux organiser le travail et répartir les taches selon les envies de chaque membre, nous avons créé un planning prévisionnel sous forme d'un diagramme de Gantt. Nous avons essayé de faire en sorte que chacun ait une tache à faire en permanence tout en respectant la durée associée pour ne pas accumuler des retards. Naturellement, nous avons toutefois rencontré des difficultés au niveau de la conception. Par exemple, nous n'avons pas eu l'occasion de faire ce projet en présentiel, ce qui aurait dû beaucoup faciliter la communication et l'entraide pour résoudre les éventuelles problèmes. Mais, grâce à la motivation de l'équipe, le projet dans son ensemble s'est bien déroulé et nous avons réussi à créer un compilateur qui répond presque parfaitement au cahier des charge fourni.

2 Organisation

L'organisation de l'équipe s'est fait assez naturellement : les envies de chacun étaient en adéquation et nous avons ainsi chacun eu la possibilité de travailler sur la partie qui nous intéressait. Ainsi deux personnes se sont chargées de l'étape A, dont une qui travaillait sur l'étape C avec un autre membre de l'équipe, les deux derniers travaillant sur l'étape B. Tout au long du projet le diagramme de GANTT était notre point de référence pour l'organisation. Cependant, nous nous appelions tout les jours à 17h pour faire un point sur ce qui a été fait durant la journée et sur l'organisation du lendemain. De plus, si l'un de nous rencontrait un gros problème ou un bug durant la journée il mettait un message sur notre groupe Discord afin d'avancer rapidement, les autres membres de l'équipe essayant de rester à disposition afin d'être le plus réactif possible.

3 Historique du projet

3.1 Ordre de conception choisi

Dès le début du projet nous avons souhaité avoir une approche incrémentale vis-à-vis du développement de notre compilateur. C'est pourquoi, nous nous sommes basés sur la réalisation de trois objectifs principaux durant le projet : faire compiler "Hello World !" ; avoir un compilateur fonctionnel pour Deca sans objet ; le compilateur doit fonctionner le mieux possible sur tout le langage Deca. En parallèle du dernier objectif, nous développions également l'extension. De plus, cette approche incrémentale a été gardée durant le développement, pour

chaque petite fonctionnalité on essayait tout d'abord de la faire fonctionner pour les cas simples, puis d'élargir à des .

3.2 Temps des différentes activités

3.2.1 Partie A

Analyse : 20%, la partie A était assez claire, il n'a pas fallu longtemps avant de comprendre et appréhender le travail à fournir. Cependant, comprendre le langage ANTLR4 a été assez compliqué vu que nous venions d'y être initié, et donc a été un peu chronophage.

Conception : 40%, après avoir su ce qu'il fallait faire, il ne restait plus qu'à comprendre comment le faire. La plupart des complétions à effectuer étaient semblables, cependant, certains objets comme le IfTheElse ont nécessité plus de réflexion quant à l'implémentation donc plus de temps que prévu.

Validation/Debugging : 35%, la phase de debugging a été assez longue, car il fallait soit régler les problèmes de Location dans les arbres, ou alors trouver les bugs et les résoudre sur un langage peu maîtrisé. Les premiers test n'étaient que peu réussi du premier coup et donc nécessitaient plusieurs tentatives supplémentaires interposés de débuggage.

Documentation : 5%, vu qu'il n'y a pas eu d'écart avec ce qui est demandé le sujet, la documentation concernant la partie syntaxique et lexicale est très peu conséquente et donc ,n'a pas pris beaucoup de temps.

3.2.2 Partie B

Analyse: 25 %, Au début, nous nous sommes laissé suffisamment de temps pour bien comprendre les notions d'environnements ainsi que pour maîtriser les différentes méthodes de récupération de données dans les classes fournies pour s'en servir dans la vérification de manière ultérieure.

Conception / Codage: 40 %, Aprés avoir commencé à appréhender les outils de travail et s'être fait une idée globale sur la vérification contextuelle, nous avons élaboré une stratégie pour mieux s'organiser et commencer à coder. Nous avons alors traité en parallèle la manipulation des environnements et l'implémentation des méthodes de vérification dans chacune des classes dans le répertoire Tree. Nous avons essayer de suivre l'évolution de la partie de génération de code afin que nous puissions faire des tests dès que possible.

Validation / Debugging: 30 %, la réalisation d'une importante couverture de tests nous a permis de rencontrer plusieurs bugs et ce n'était pas toujours

assez évident de trouver la solution directement ce qui nous a parfois mis devant l'obligation de changer la structure des méthodes.

Documentation: 5 %, nous avons préféré laisser la documentation à la fin, ce qui nous permet de mieux nous concentrer sur la conception et la réalisation du compilateur.

3.2.3 Partie C

Analyse : 35 %. Durant chacune des 3 phases qui ont découpé notre projet, un temps assez considérable consistait à comprendre où était les parties à modifier dans le projet et surtout comprendre la logique derrière l'assembleur pour l'implémenter par la suite.

Conception : 30 %. Après une conséquente partie d'analyse, lorsque nous considérions avoir assez bien compris ce que nous devions réaliser, nous nous occupions de la mise en pratique.

Validation/ Debugging : 30 %. Bien sûr, nous ne réussissions pas toujours à obtenir le résultat attendu après les premiers tests (très rarement d'ailleurs). Nous essayons alors, en parallèle, de corriger nos erreurs et d'aggrandir la couverture de tests pour essayer de ne passer à côté d'aucune bévue de notre part.

Documentation : 5 % .Nous avons préféré nous concentrer sur la conception en elle-même, la documentation étant laissée dans les moments de creux ou pour la fin.

3.2.4 Extension TRIGO

Analyse : 35%, le temps d'analyse a été principalment consacrée à la recherche d'algorithme de calcul pour les méthodes demandées dans l'extension TRIGO. Avec la liberté que nous avions, nous avons pris le temps de rechercher et comparer plusieurs méthodes de calcul des fonctions à implémenter, ce qui a pris environ un quart du temps du travail total. Aussi, les contraintes de Deca non présentes sur Java ont demandées un tant de réflexion supplémentaire afin d'être contournées.

Conception : 25%, la conception a pris quelques temps du fait que lorsque nous travaillions sur l'extension, le compilateur n'était pas achevé, nous avons donc commencé par tout implémenter en Java, vérifier que tout fonctionnait bien, puis assurer une "traduction" en deca.

Validation/Debugging : 10%, étant donné qu'il s'agissait dans la plupart des cas, d'algorithmes comportant des guides, peu d'erreurs subsistaient et donc

le débuggage était très rapide.

Documentation : 30%, la documentation de l'extension à réaliser est assez longue. Le temps de faire des tests supplémentaires de performances, de précision ou autre est également assez conséquent, tout comme réunir toutes les informations vues pendant l'analyse et la conception.

4 Conclusion

Finalement, le projet GL était clairement très formateur pour tous les membres de l'équipe. C'était une bonne occasion de participer à un projet dont la forme se rapproche de ce que nous pourrions retrouver en entreprise. La motivation, la rigueur et un bonne organisation d'équipe nous a permis d'atteindre notre objectif : parvenir à délivrer notre compilateur complet respectant les spécifications dans les délais accordés.