Thomas Cassagnes

Bilan individuel

Ce document explique en quoi j'ai progressé sur deux des cinq compétences indispensables à la bonne réalisation d'un projet informatique en équipe tel que le compilateur Deca. En effet, le projet devait répondre aux exigences d'un client, avec un cahier des charges et des deadlines régulièrement, demandant ainsi une certaine organisation et un travail en équipe efficace. Les codes implémentés devaient être couvert par une base de tests, afin de produire un programme fonctionnel et de qualité. La pertinence du rendu devait être assuré par la maîtrise technique des différents domaines informatiques mis en jeu par le projet.

Compétence 1 : Travailler en mode projet

Ayant déjà eu l'occasion de travailler en équipe nous avons immédiatement pris le temps de se fixer des règles de travail communes, sachant que ce temps de concertation nous permettrait de ne pas être freiné et d'en perdre au fur et à mesure que nous avancerions dans le projet. Des horaires fixes de travail ont été établies, nous travaillions aussi à côté les uns des autres, afin de créer une certaine émulation et de pouvoir s'entraider facilement pour rester dans l'avancée. Nous fixions de même un Scrum Talk par demi-journée, pour connaître l'avancée, les problèmes et les futures tâches de chacun au sein de l'équipe.

Les compétences et envies de chacun furent prises en compte dans un premier temps afin de se répartir le travail, le retard que nous avons pris lors des premiers jours nous a toutefois obligé à se distribuer les tâches suivantes de manières à respecter les deadlines les plus urgentes. De plus, nous avons alors repensé notre utilisation des méthodes agiles et notre planning prévisionnel (peu efficace en raison de notre faible connaissance des tâches à réaliser). Nous avons créé un projet Trello sur lequel les différentes catégories apparaissaient :

- « A faire », toutes les tâches à réaliser nous venant pour le moment à l'esprit, en essayant de les noter progressivement au fur et à mesure que nous passions d'une deadline à une autre. Nous y voyons ainsi plus clair quant à notre avancement pour tel deadline.
- « En train de faire », les tâches en cours d'exécution, pour savoir qui travaille sur quoi.
- « Code review », chaque portion de code fut relu par un membre de l'équipe autre que son écrivain, l'utilité de travailler côte à côte fut alors encore démontré.
- « A tester », les méthodes complétées et passées par la catégorie « Code review », mais par encore testées, pour lesquelles il nous fallait étendre notre base de tests.
- « Fait », recueillant l'ensemble des tâches finies, relues et testées. Cette catégorie nous permettait de mesurer notre avancement, ainsi que de se motiver devant le travail déjà effectué.
- « A déboguer », les méthodes implémentées mais sur lesquelles il subsiste des erreurs que l'écrivain n'est pas parvenu à éliminer.

J'ai donc découvert le potentiel d'un projet détaillé sous Trello, permettant de ne pas perdre de vu les objectifs et attentes du travail.

Ayant commencé par développer le lexer et le parser, j'ai été surpris lors de la fin de la réalisation de la partie Sans Objet par l'étendue des méthodes et tâches que les méthodes agiles nous permettaient alors d'implémenter. Celles-ci nous apportaient aussi satisfaction et donc motivation lorsque le compilateur put enfin compiler des sous-langages de Deca. Elles nous ont cependant fait défaut lors de la partie Objet de l'étape C, où la première implémentation orientée Sans Objet ne convenait plus et rendait impossible celle demandée sans reprendre le code. Ces méthodes de gestion de projet m'auront tout de même prouver leur efficacité sur un projet qui est bien moins accessible abordé en blocs.

L’organisation finalement assez complète de notre groupe m’a procuré un encadrement efficient pour coder dans de bonnes conditions et pour profiter des avantages du travail en équipe.

Compétence 2 : Mettre en œuvre des processus de validation

Il est essentiel pour un projet implémenté pour un client que les différentes fonctionnalités soient opérationnelles et qu’elles répondent de manière pertinente à toutes éventualités. C’est dans cet objectif que nous avons travaillé et que nous avons développé notre base de tests. J’ai été étonné par la quantité de tests nécessaires pour vérifier les problèmes auxquels peut faire face chacune des étapes de l’implémentation d’un compilateur. En effet, l’élaboration continue de nos tests nous a permis de détecter des erreurs sur des méthodes conçues fraîchement mais aussi au début du projet.

Devant l’étendue de la base de tests, nous avons dû construire des scripts pour lancer automatiquement les tests et pour comparer les résultats du code avec les attentes du cahier des charges (pour certaines parties du projet où c’était pertinent). Des modifications ultérieures pouvaient en effet affecter la validation antérieure d’un programme. J’ai par exemple écrit un script comparant les arbres créés par le parser et ceux attendus par la grammaire du langage Deca. Il nous a alors permis de repérer une erreur sur l’affichage des chaînes de caractères dans l’arbre non décoré, peu gênante à priori, mais qui générer de nouvelles erreurs dans l’étape C de génération de code ima.

L’outil Cobertura s’est révélé très efficace pour déceler des portions de code non exécutées par nos tests, au début très nombreuses (seulement 40 % de couverture) comme cela à pu me surprendre, puis raisonnablement peu (78 % de couverture) à mesure que nous étoffions notre base de tests malgré l’ajout de lignes de code. Pour lancer Cobertura, nous avons finalement automatisé le lancement de l’ensemble de nos tests, avec l’aide du script all-test implémenté dans le dossier src/test/script/.

Ayant beaucoup travaillé sur l’extension (TRIGO), j’ai constitué une batterie d’outils pour tester selon différents aspects les algorithmes que j’ai pu implémenter pour calculer les fonctions de l’extension.