Bilan de gestion d’équipe

1 Description de l’organisation

Au départ du projet, l’objectif était que chaque membre puisse travailler sur au moins deux étapes du projet, parmi les étapes A, B, C et l’extension, le but étant que chacun puisse avoir une compréhension du projet la plus complète possible. Cependant, le manque d’expérience dans la gestion et la réalisation de projets, a fait qu’il n’a pas été possible de mener à bien ces objectifs. Au sein de l’équipe, seuls Najète et Rémi ont réellement travaillé sur deux étapes parmi les quatre citées, en l’occurrence l’étape A et l’étape C. A l’inverse, Hicham et Oussama se sont essentiellement concentrés sur l’extension trigonométrique, après avoir travaillé sur le début de la partie B (sans objet) et Alexian a essentiellement travaillé sur l’intégralité de l’étape B, en plus de la mise en place des scripts de test.

Malgré des membres de l’équipe travaillant sur des parties différentes pendant la totalité du projet, l’équipe a fait en sorte que la communication soit une dimension importante du projet. Chacun était à peu près au courant de l’avancement des autres membres pendant les trois semaines du projet, et la communication a également été d’une grande aide pour déceler les bug ou les cas non traités dans les parties des autres membres.

Nous tenions à ce que chacun des membres puisse donner son avis ainsi que son ressenti vis à vis de l’organisation de l’équipe pendant le projet dans ce bilan. Vous trouverez ces avis ci- dessous.

Alexian : Dans l’ensemble, l’organisation de l’équipe n’a pas été optimale, la gestion du temps de certaines fonctionnalités a souvent été sous estimée, créant des retards dans une étape, bloquant temporairement l’étape qui la suit. Cette mauvaise estimation des délais a peut être été causée par un manque de connaissance vis à vis du travail à réaliser. L’équipe a travaillé essentiellement en équipe, surtout sur les parties A, B et C où la communication a permis de régler une grande partie des problèmes rencontrés.

Hicham : Globalement, la répartition des tâches n'a pas été parfaite et ceci est dû au fait qu'au début, on avait du mal à estimer la durée des tâches comme il le fallait, ce qui a fait que la plupart du temps, on avait sous estimé le volume horaire à

consacrer à nos tâches. Cependant, la communication a été un des atouts de notre groupe et nous a permis d'exposer nos problèmes et de nous aider à les surmonter.

Oussama : La gestion du projet a rencontré quelque problèmes dus à une mauvaise estimation du travail nécessaire pour certaines parties, ce qui causait des décalages avec l'avancement du projet. La connaissance croissante des parties demandées acquise au cours du projet et la bonne communication entre les membres du groupe ont permis de surmonter la majorité de ces difficultés pour avoir une fin de projet meilleure que son début en terme de gestion des tâches.

Najète : J'ai personnellement trouvé que se séparer en différentes équipes était une stratégie intéressante, et que ce qui a vraiment manqué pour un rendu parfait était une compréhension générale plus rapide du projet la première semaine. La communication a aidé à résoudre de nombreux bug.

Rémi : J'ai bien apprécié notre répartition globale des tâches, car pouvoir me focaliser uniquement sur ma partie du travail en faisant confiance à mes collègues sur les leurs m'a permis de vraiment me concentrer sur mon travail et de gagner beaucoup en efficacité. Le groupe était toujours très réactif au moindre signalement de bug, notamment grâce à une communication efficace.

2 Historique du projet

- Ordre d'exécution

L'exécution du projet a été conforme au polycopié. L'étape A, comportant le lexer ainsi que le parser, a toujours été réalisée en premier, que ce soit pour la partie sans objet ou celle avec. Pour essayer d'être dans les temps, le parser avec objet a été commencé et fini avant le rendu intermédiaire afin de permettre aux étapes suivantes de bien commencer. L'étape B, s'appuyant sur la A, pouvait alors commencer et a notamment permis de remonter des bug au niveau des Location, concept qui n'avait pas été bien compris au départ. Enfin, l'étape C venait clore le compilateur et a, pour sa part, permis de découvrir des problèmes persistants dans les parties précédentes comme les différentes définitions.

- Etape B

L'étape B, se composant de vérifications contextuelles et décorations de l'arbre, a été abordée avec l'objectif de produire et d'implémenter des fonctionnalités dans un ordre cohérent, avec en priorité les parties essentielles pour la poursuite du

projet. On a tout d'abord commencé par une prise en main du code, avec lecture de la documentation et lecture du code associé pour connaître les enjeux de cette étape. Dans le même temps, pour la prise en main, on a réalisé l'implémentation de la partie utilisée pour compiler le programme *"Hello World"*. Par la suite, l'objectif était de produire la partie sans objet pour le rendu intermédiaire. Pour cela, des objectifs ont été fixés, avec en priorité les éléments du code essentiels pour d'autres fonctionnalités. Ces objectifs ont été, dans l'ordre, l'établissement d'un environnement pour les variables, puis la déclaration de variables (gestion des doubles déclarations aussi) sans puis avec initialisation, opérations arithmétiques puis opération de comparaisons. Ces différentes fonctionnalitées étaient nécessaires à la mise en place et à la validation des fonctionnalités telles que les conditions *"if then else"* et *"while"*, qui ont été implémentées par la suite.

Concernant la partie objet, l'ordre de conception s'est fait en fonction de l'ordre des différentes passes et de la dépendance des fonctionnalités les unes avec les autres. Le développement s'est fait en priorité sur la première passe, avec la mise en place de l'environnement des classes, la déclaration de classes sans extension puis avec extension. Puis pour la seconde passe, le développement a continué avec, dans l'ordre, la déclaration de champs, avec et sans initialisation et visibilité, et la déclaration des en-têtes des méthodes avec leur signature. Finalement, pour la dernière passe, concernant le corps des méthodes ainsi que le programme principal (main), il a fallu, dans un premier temps, gérer l'environnement du corps des méthodes pour les visibilités des différentes variables. Pour ce qui est de l'implémentation des déclarations de variables et la liste d'instructions, cela a été proche de ce qui avait déjà été fait lors de la partie sans objet. Tandis que pour le programme principal, de nouvelles fonctionnalités ont été prises en charge, comme les appels de méthodes dans la classe et à partir de l'objet, la surcharge de méthode. Contrairement à ce qui était prévu, la gestion de l'instruction d'appartenance *"instanceof"* a pu être implémentée dans l'étape B, étant donné que les classes concernées avaient été créées lors de l'étape A avec objet.

- Etape C

L'étape C consiste essentiellement en la génération de code en assembleur. Afin d'aborder cette partie, une première partie du travail a consisté à étudier les exemples à notre disposition ainsi que le squelette de code afin de comprendre où se rajouteront les instructions et leur logique. On a alors créé la classe Codegen où les méthodes permettent de générer le code assembleur propre à différentes actions, telles que le chargement de valeur dans un registre à l'aide d'un gestionnaire de registres. Pour la partie sans objet, les principales fonctionnalités mises en place ont été, dans l'ordre, la déclaration de variables, avec et sans initialisation, suivie de la (ré)assignation de variables de différents types. S'en est

suivie l'implémentation des opérations arithmétiques, accompagnée de la gestion potentielle des flottants, avant que l'on ne s'attaque aux opérations booléennes, en commençant par les opérations de comparaison avant de faire les opérateurs logiques. Cela nous a alors permis de gérer les opérations conditionnelles (If_then_else et while).

En ce qui concerne la partie objet, nous avons pris le temps de bien s'imprégner des différentes facettes et subtilités du code assembleur qui accompagne les classes (construction de la table des méthodes, code des méthodes). Cela a été plus compliqué que pour la partie précédente car il fallait coder une partie assez importante avant de pouvoir mener de vrais test. Nous avons commencé par créer un squelette rudimentaire nous permettant de coder des classes basiques, en reprenant notre approche incrémentielle. La première étape a été la construction de la table des méthodes, qui gère également les méthodes héritées. Ensuite, on a implémenté la sélection de champs, ainsi que l'initialisation d'un élément d'une classe avant de s'attaquer au code des méthodes et à leur appel.

- Extension

Le travail sur l'extension, beaucoup moins guidé que les parties décrites jusqu'à présent, a consisté, dans un premier temps, en une prise de conscience des différentes fonctions à implémenter et en une analyse des dépendances entre les différentes fonctions ainsi que les éventuelles réductions d'intervalles. S'en est suivie une recherche des algorithmes possibles pour l'approximation, comme les algorithmes de Taylor, Cordic, Weierstrass et de ceux pour les fonctions intermédiaires comme la fonction racine carrée ainsi que la fonction puissance, à savoir l'algorithme d'exponentiation rapide ou encore l'algorithme d'Héron… Cette étape nous a permis d'avoir une vision plus claire du travail demandé. Ensuite, nous nous sommes attaqués à l'implémentation des premières versions naïves mais fonctionnelles de sinus et de l'arc-tangente qui offrent la possibilité de tester leur niveau d'approximation et de découvrir les points critiques où les fonctions sont moins précises. Des tests et des implémentations supplémentaires ont servi à augmenter la précision des fonctions sinus et arc-tangente ainsi que la correction des bug. Nous sommes ensuite passés à l'implémentation des fonction cosinus, arc-sinus et arc-cosinus. Le travail s'est terminé avec une implémentation de la réduction d'angle pour le sinus et le cosinus et une comparaison avec les fonctions trigonométriques du Java.

- Temps passé sur les différentes activités

Au cours du projet, la gestion du temps a été un point crucial dans la réalisation du compilateur. Tout d'abord, il nous a fallu prendre connaissance du projet et donc du code fourni. Cette analyse de code a duré essentiellement la première semaine, au commencement du projet génie logiciel. En ce qui concerne la documentation à produire pour le rendu final, il y a eu un début de rédaction au cours de la deuxième semaine, avec une vitesse de progression plutôt lente jusqu'au week end précédent le rendu final, à partir duquel un sprint de la rédaction s'est mis en place. Un investissement plus anticipé dans la rédaction des différentes documentations aurait pu réduire l'intensité du sprint. Pour valider notre programme, des jeux de tests ont été produits, la production des scripts a pris environ deux jours. Après quoi, il fallait uniquement produire des fichiers *".deca"* pour nourrir les tests. Cette phase de production de test s'est faite de la deuxième semaine jusqu'au rendu final, en parallèle avec la production de code. Un plus grand nombre de fichiers et donc de cas traités aurait pu augmenter la fiabilité de notre programme, le temps investi dans la création de test aurait pu être meilleur. Le code, quant à lui, a représenté la majorité du temps des différents membres de l'équipe. Pour Najète et Rémi avec l'étape C, Hicham et Oussama lorsqu'ils travaillaient au début de l'étape B, après quoi la majorité de leur temps était de la recherche concernant l'extension. Alexian a passé la majorité de son temps à écrire du code pour l'étape B. A partir de la deuxième semaine jusqu'au week end précédent le jour du rendu final, la programmation a monopolisé la majorité de notre temps sur le projet. Le temps passé sur la partie code a été trop importante, cela a mobilisé la très grande majorité du temps des membres de l'équipe, réduisant le temps pour les autres parties du projet, qui étaient tout aussi importante.

Si l'on devait résumer, notre temps s'est donc réparti de la manière suivante :

- Analyse du code → la première semaine.

- Documentation → début milieu de la deuxième semaine avec une avancée lente jusqu'au rendu après quoi sprint de la rédaction.

- Conception → touchant essentiellement la partie C où la classe Codegen a été créée de toute pièce. La durée peut être estimée à trois jours, au début de la deuxième semaine.

- Codage → Le code a représenté la majorité du temps de Najète, Rémi et Alexian. Pour Hicham et Oussama c'était de même durant les deux premières semaines (travail dans la partie B) et puis consécration de plus de temps pour des recherches concernant les algorithmes d'approximation, les flottants etc .

- Validation → La conception des scripts de tests a pris environ deux jours, quant à l'ajout de fichier/cas à tester, ils ont été ajouté en parallèle de l'implémentation de chacune des fonctionnalités et donc le temps de leur mise en place a été étiré sur les deux dernières semaines.