Bilan de l'organisation du projet

Pour l'organisation générale de l'équipe, je vous renvoie vers la Charte d'équipe éditée en début de projet et présente dans le même répertoire *./docs*. Ce document analysera plus en détails la répartition, indiquera les choix pris au fur et à mesure du projet (en dynamique), et présentera les critiques à posteri par rapport à ces décisions.

1. Description critique de l'organisation adoptée dans l'équipe.

Le projet peut être découpé en 3 grosses étapes, qui constituent les points 1.1, 1.2 et 1.3. Chacun de ces points représentent une semaine de travail.

1.1. Langage Hello-World

Le langage Hello-World est le point d'entrée du projet, il s'agit d'un langage très simple qui permet d'afficher des chaînes de caractères grâce à une fonction print. Ce premier langage permet de s'investir dans la compréhension de la structure du compilateur sans devoir dominer des aspects plus techniques nécessaires par la suite. Il s'agit du minimum à atteindre pour considérer que la première semaine de développement a bien été aboutie.

1.1.1. Organisation des rôles et de l'équipe

À cause des contraintes extérieures affectant deux membres du groupe, les deux premiers jours du projet se sont déroulés en distanciel pour notre équipe. Nous avons considéré que cela ne serait pas spécialement contraignant puisque nous devions regarder des vidéos de cours d'introduction au projet. Par conséquent nous avons réalisé notre première réunion de suivi et d'organisation le mercredi 5 janvier. Lors de cette première réunion nous avons décidé de la répartition des rôles au sein de l'équipe : un chef de projet (Nathan) désigné pour gérer en priorité les aspects de gestion de projet et le reste de l'équipe sur un pied d'égalité serait assignée à des rôles précis selon le besoin.

Cette organisation plutôt vague nous donnait des libertés qui à notre avis convenaient bien à une équipe très homogène comme la nôtre. Ensuite, à partir des informations que l'on avait retenu pendant les premiers jours, nous avons mis en place un Gantt avec lequel nous avons structuré un planning prévisionnel. Finalement au cours des 2 derniers jours avant la réunion de suivi du vendredi, nous avons fait travailler 2 personnes sur la partie A et 3 personnes sur la partie B ce qui nous a permis de terminer jeudi soir puisque la partie C était négligeable.

1.1.2. Stratégie de test et mise au point

Comme expliqué précédemment le langage Hello-World est la prise en main du projet, il permet une première approche très simple techniquement mais qui donne une première vision générale sur le fonctionnement du compilateur. Par conséquent, il était compliqué de déterminer des tests approfondis dès le début, on a donc décidé d'utiliser les tests fournis avec le code. Tout en ajoutant quelques tests complémentaires pour les fonctionnalités telles que le include (fonctionnalité qui permet de lire le code sur plusieurs fichiers). Notre organisation pour ce sprint a été très simple par la suite, mise en place du code d'une étape, puis vérification avec les tests fournis pendant que les autres démarraient l'étape suivante. Cette organisation s'est révélée largement suffisante pour ce premier sprint, puisque nous avons mis en place le langage et assuré son fonctionnement dès jeudi soir. En conclusion, la première semaine s'est déroulée comme prévu, nous avons mis en fonctionnement le langage Hello-World pour le présenter vendredi matin lors du premier suivi de projet. Et de l'approfondir en faisant des affichages d'entiers et flottants pendant la journée de vendredi.

1.1.3. Retour critique sur le temps passé

À la fin de chacun des sprints nous avons fait une révision des positifs et négatifs de la gestion et la réalisation du projet. Pour ce premier sprint nous avons relevé deux grands aspects positifs : la répartition du travail entre les membres de l'équipe et le respect des objectifs du planning. Puis nous avons relevé deux points négatifs : le manque de tests personnels, aspect que l'on a amélioré par la suite, et un léger manque de rigueur. En effet nous avons compris ce premier langage, mais nous n'avons pas respecté au pied de la lettre la description de la syntaxe pour son implémentation. Nous avons donc corrigé cet aspect pendant le week-end, notamment en adaptant le Lexer à la même syntaxe que celle décrite sur le polycopié. D'autre part, tout au long du projet, nous avons pu réfléchir davantage à nos sprints et nous avons pu approfondir les positifs et négatifs de notre gestion de projet. Par exemple, nous considérons aujourd'hui que le langage Hello-World aurait dû être fini encore plus tôt, à travers un investissement plus grand dans l'implémentation durant les deux premiers jours. Nous considérons que nous aurions été plus efficace en rentrant directement dansle code et en passant moins de temps sur la documentation.

1.2. Langage Sans Objet

Le langage sans objet est le deuxième langage du compilateur Deca, à implémenter pour le rendu intermédiaire. Il étend énormément le langage précédent puisqu'il permet déjà d'écrire des fonctions plus complexes, en effet il permet d'initialiser des variables, les utiliser pour des opérations et implémente des structures de contrôle tel que des if et des boucles while. Nous avons commencé son implémentation le vendredi de la première semaine après notre suivi de projet et l'avons terminé la veille du rendu intermédiaire.

1.2.1. Organisation des rôles et de l'équipe

Nous avons commencé ce sprint en appliquant ce que nous avions appris lors du sprint précédent et lors de la séance de suivi. Nous avons mis deux personnes sur la réalisation de tests et l'automatisation pour pouvoir tester de façon complète au fur et à mesure du développement. Puis nous avons distribué le reste des tâches entre les 3 membres restants. Nous avons fait davantage attention à les diviser plus clairement. Par exemple, nous avons divisé l'étape B en plusieurs sous-phases disjointes pour pouvoir les répartir entre tous les membres et avancer plus rapidement.

D'autre part, nous avons fait l'effort de travailler tous ensemble à l'Ensimag, ce qui nous a permis d'être tous investis dans l'ensemble du projet. Effectivement, cela nous permettait d'échanger à plusieurs reprises pendant la journée, pour faire des points sur l'avancement de chacun des membres.

1.2.2. Stratégie de test et mise au point

Comme décrit lors du paragraphe précédent, nous avons compris que les tests allaient être une partie essentielle du développement, qui nous permettrait de rattraper rapidement toutes nos erreurs. Par conséquent nous avons utilisé les tests fournis comme base et avons écris davantage de tests. Pour cela, nous avons décidé de séparer les personnes qui réalisent les tests de celles qui implémentent le code, les tests étaient donc réalisés uniquement par rapport à la description donnée sur les polycopiés du projet. Ce fonctionnement nous a permis de rattraper des erreurs de compréhension, notamment au niveau du Lexer et du Parser.

La mise au point s'est déroulée plutôt bien, du fait que l'on a atteint l'objectif du rendu en bonnes conditions. En effet nous avions un compilateur fonctionnel sur ce langage dès le vendredi, et nous avons pu utiliser notre week-end pour corriger des bugs et optimiser l'implémentation.

1.2.3. Retour critique sur le temps passé

Ce deuxième sprint s'est révélé plus compliqué que le premier, non seulement d'un point de vue technique mais aussi d'un point de vue de l'organisation. Mis en confiance par nos bons résultats lors du premier sprint, nous avons commencé la semaine en pensant que la suite serait facile. Nous nous sommes rendus compte par la suite que nous avions sous-estimé la charge de travail. Par conséquent nous avons réalisé des journées moins chargées les premiers jours de la semaine et avons dû compenser par du travail plus intense en fin de semaine et pendant le week-end. Nous avons quand même continué à respecter notre planning initial, mais nous avons dû faire des longues journées en fin de semaine pour assurer un premier rendu de qualité correctement testé.

C'est aussi durant ce sprint que nous avons développé un système complet d'automatisation des tests, car nous nous sommes rendus compte lors du 1er de

l'importance de celui-ci. Ce système nous a fait gagné énormément de temps par la suite.

1.3. Langage Complet et Extension

Cette 3ème semaine a été la plus intense du projet. Malheureusement, à ce stade nous avions que très peu avancé l'extension car nous avions dû concentrer notre énergie sur le rendu intermédiaire.

1.3.1. Organisation des rôles et de l'équipe

Nous avons ainsi décidé de mettre deux personnes à temps plein sur l'extension, deux personnes sur le développement du langage complet, et une personne sur l'écriture des tests pour le langage complet. Il ne fallait pas non plus négliger l'intégration de l'extension dans le langage avec objet, à la fin du développement. Les deux binômes constituées étaient autonomes pour gérer leur répartition du travail, qui comprend aussi la décision de travailler en pair programming sur certains points clés.

1.3.2. Stratégie de test et mise au point

Comme expliqué sur le polycopié, il était nécessaire que les résultats des tests du langage sans objet soient les mêmes à la suite du développement du langage avec objet. C'est pourquoi les sorties avaient été stockées dans des .lis et "mvn test" permettaient de vérifier régulièrement qu'aucun résultat n'avait changé. Pour les tests du langage complet, la même stratégie a été adoptée. Des tests sont écrits en parallèle du développement et sont ainsi lancés à la fin de chaque étape (A, B et C). Ils permettent d'ajuster le développement.

Pour l'extension, le développement des tests était un peu différent car les spécifications n'étaient pas déjà existantes. Ainsi, les personnes qui ont réalisé les tests sont les mêmes personnes qui ont développé l'extension.

1.3.3. Retour critique sur le temps passé

Comme dit en introduction, cela a clairement été la semaine la plus intense du projet. Mais je pense aussi que cela a été la semaine la plus productive. Les tâches étaient très bien réparties et la part alloué à chaque étape (analyse, conception, codage, validation) a été bonne. Les deux semaines précédentes ont aussi permis à l'ensemble des membres d'avoir plus de recul sur le projet et ainsi d'avancer plus efficacement.

2. Points de vue sur le cadre fixé et le projet

Le Projet Génie Logiciel est une étape très enrichissante dans la formation à l'Ensimag. Il nous permet de travailler en équipe, sur un projet qui dure dans le temps, et ainsi nous former à notre futur métier d'ingénieur. L'implémentation

d'un compilateur permet de progresser d'un point de vue technique en programmation, mais aussi de prendre du recul sur des outils que nous utilisons quotidiennement.

2.1 Formation des équipes

Nous sommes tous d'accord pour dire que les contraintes sur la formation des équipes est une chose positive. Cela permet d'élargir le champ des compétences de l'équipe de façon globale. Même si bien évidemment, une équipe de Projet GL reste très homogène (même âge, même niveau d'étude, même formation, etc.) par rapport à une équipe en entreprise.

2.2 Un apprentissage sur le "tas"

Même si le projet est globalement guidé, certaines parties nécessitent de la prise d'initiative. L'automatisation des tests en est la meilleure illustration. Aucune formation nous a été fourni et c'était pour nous une première expérience en la matière. Cela nous a formé sur cette notion, mais surtout nous avons dû nous former nous-même. Apprendre, s'adapter à son environnement et monter en compétences par soi-même sont des capacités nécessaires au métier d'ingénieur. C'est pourquoi il est important de nous laisser nous débrouiller sur certains points.

2.3 Pas de documentation fournie sur les classes et le code existant

Le fait de ne pas fournir de documentation sur le code existant est quelque chose d'étonnant. Il faut comprendre les classes fournies et faire le lien avec les spécifications du polycopié. Il y a des héritage sur plusieurs niveaux et cela rend la compréhension encore plus difficile. Un simple diagramme UML des classes existantes aurait été bien. Cela nous aurait permis de comprendre plus rapidement et de dégager du temps pour réaliser un compilateur d'encore meilleur qualité, ce qui aurait été je pense plus enrichissant.

3. Conclusion

Etant une équipe motivée et consciente des enjeux de la gestion de projet, nous avons pu développer de façon relativement efficace le compilateur Deca ainsi que l'extension que nous avions choisi. Cependant, à posteri, nous considérons qu'au départ nous n'avons pas suffisamment réparti les tâches. Notamment, nous aurions dû dès le début mettre au moins une personne sur l'étape C: la génération du code, car même si elle est dépendante de l'arbre construit aux étapes précédentes, elle nécessite une compréhension des instructions assembleurs et ceci peut se faire indépendamment. Cette répartition non parfaite est aussi dû au manque de connaissances techniques et de recul que nous avions sur le projet.