Gestion d'équipe et de projet

GL28

Janvier 2020

Table des matières

1 Description critique de l'organisation

1.1 Introduction

La variété des tâches à réaliser durant la conception du compilateur (développements étape A/B/C, extension, tests...) a imposée une répartition des tâches dans l'équipe. Lors de la première semaine, nous avons constatés la charge de travail exigée pour la réalisation du projet et nous nous sommes entendus pour réaliser un compilateur objet et non la version complète. Un point notable dans notre organisation est le fait que nous avons chaque semaine travaillés sur un nouveau sous ensemble du langage Deca (semaine 1 : Hello world, semaine 2 : sans objet, semaine 3 : objet). Il s'agit de la méthode de travail AGILE, nous avons procédés par itérations successives. Cependant notre manque d'expérience a eu pour conséquence une période creuse en début de semaine 1, 2 et 3 : à chaque début de semaine, il fallait avancer l'étape A avant de pouvoir réaliser concrètement l'étape B et C. Notre organisation aurait gagnée à mieux anticiper ces moments pour pas accumuler de retard. Mais il convient de souligner que notre planification a été relativement juste et respectée, cependant nos rôles dans l'équipe ont beaucoup variés en semaine 1 et se sont fixés en semaine 2 et 3.

1.2 Répartition des rôles

1.2.1 Semaine 1

Comme dit précédemment, nous n'avions pas vraiment de rôles attribués lors de la première semaine, et chacun essayait de comprendre les différentes attentes du compilateur afin d'avoir un vue d'ensemble convenable du projet. A posteriori, un manque d'investissement dès le début de la semaine fut en partie responsable de la faible dynamique du groupe sur la semaine 1.

1.2.2 Semaine 2

— 1 personne 'dev exclusif' sur l'étape A et développement de tests
— 1 personne 'dev exclusif' sur l'étape B
— 1 personne 'dev exclusif' sur l'étape C
— 1 personne en charge des tests
— 1 personne responsable de l'extension

La principale amélioration en terme d'organisation est l'affectation d'un membre de l'équipe aux tests à plein temps. En effet nous avions noté une lacune dans ce domaine après le suivi n1 - le choix de procéder en mode TDD (Test Driven Development) nous a permis d'obtenir un compilateur sans objet satisfaisant au rendu intermédiaire malgré notre retard.

1.2.3 Semaine 3

— 1 personne 'dev exclusif' sur l'étape B
— 1 personne 'dev exclusif' sur l'étape C
— 1 personne 'dev exclusif' sur l'étape A puis responsable des tests partie objet
— 2 personnes sur l'extension

La mise au point de l'extension **TRIGO** ayant peu avancée jusque là, nous avons augmenté le nombre de ressources attribuées afin d'obtenir des résultats à la hauteur de nos attentes. Cela fut possible car nous étions de plus en plus efficace avec la méthode TDD : en effet la roadmap de la partie objet fut la validation des tests pré-existants, des plus faciles aux plus délicats.

1.3 Horaires

Nous avons assez vite remarqué qu'imposer des horaires de travail à l'imag ne serait pas compatible avec les préférences de chacun. Ainsi nos horaires étaient flexibles : arrivée de 7h30 jusqu'à 11h, sortie à partir 16h00 jusqu'à 21h00, travail à distance avec ssh, parfois la nuit .. La communication sur notre serveur Discord a pallié à ce déphasage du travail.

1.4 Communication interne

En début de projet, nous avons mis en place un serveur Discord et un Trello afin de communiquer notre avancement, nos remarques, constituer une roadmap etc .. En divisant ce serveur Discord en plusieurs channels, nous avons vite réalisé que le Trello était redondant et l'avons donc abandonné. Ces *channels* étaient réparties dans des **catégories** :

- **Organisation**
  - *général*
  - *planning*
  - *suivi et soutenance*
  - *todo*
- **Dev**
  - *étape A*
  - *étape B*
  - *étape C*
  - *extension*
  - *tests*
  - *bugs*

2 Présentation de l'historique du projet

2.1 Git

Dans un premier temps, nous utilisions une branche par fonctionnalité et cela causait beaucoup de problèmes, notamment car nos rôles étaient mal définis, et l'on perdait du temps à mettre en commun notre travail. Lors de la semaine 2, nous avons mis en place un *workflow git* qui s'est révélé efficace (voir illustration).

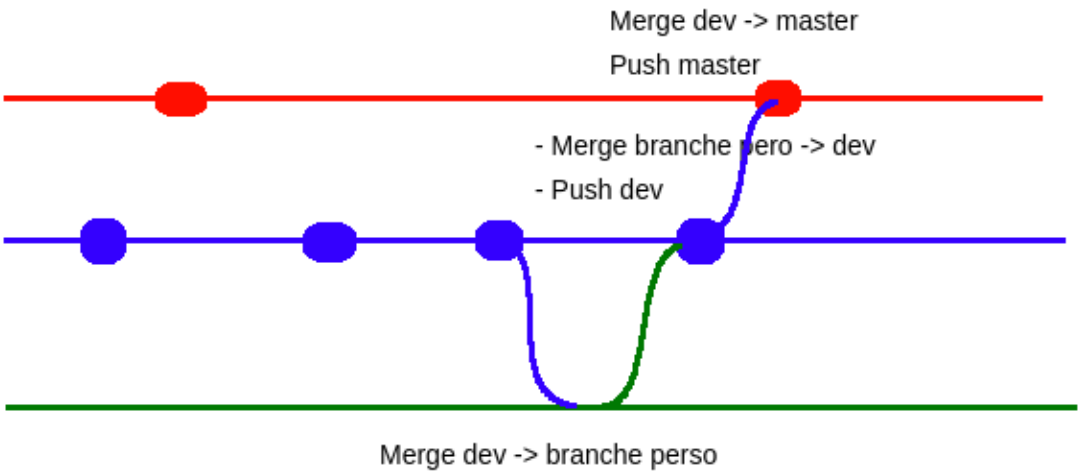

FIGURE 1 – branche perso (vert) - dev (bleu) - master (rouge)

2.2 Conception et développement

L'ordre d'exécution du compilateur pouvant se résumer à "Étape A → Étape B → Étape C", un mode de développement "naïve" serait de coder dans cet ordre les différentes fonctionnalités du compilateur. Lors de ce projet, nous avons procédé de la manière suivante :

- Analyse et conception en commun entre étape A et B (notamment pour les classes Java des terminaux du langage objet)
- Développement en parallèle de A et des constructeurs des classes de B
- Analyse et conception en commun entre étape B et C sur les règles du langage Deca
- Développement en parallèle de B et C

2.3 Validation

Concernant la validation, nous avons fait l'erreur en semaine 1 de négliger ce point. Après avoir eu l'occasion d'échanger avec les enseignants durant le premier suivi, nous avons pris conscience de l'importance de mobiliser des ressources pour la validation de notre compilateur. La partie validation a donc pris une grande importance à partir de la semaine 2. Cette étape est très importante pour pouvoir développer un compilateur qui accepte tout les programmes valides et refuse tout les programmes invalides. Les tests développés ont étés utiles pour le développement de l'étape B et C pour pouvoir détecter le plus vite possible un maximum d'erreur de développement. Au début de chaque sprint de développement, les tests étaient en avance sur le développement, mais en fin de sprint un temps conséquent était consacré à faire passer les tests au code développé. D'un point de vu rétrospectif, cette gestion pour la validation du compilateur a été un succès, le fait de dissocier le travail de validation du travail de développement nous a permis de détecter davantage d'erreurs dans le compilateur qu'en mêlant les 2 rôles comme en semaine 1.