# Bilan de gestion d'équipe et de projet

G15 projet gl

January 25, 2023

## Contents



# 1 Méthode suivie

En plus d'avoir été pour chacun d'entre nous le premier projet de développement informatique de cette envergure, le projet Génie Logiciel à également été notre première approche des méthodes de gestion de projet couramment utilisées dans l'industrie pour le développement informatique en général. Dans ce cadre nous nous sommes orientés vers l'application de la méthode Agile.

## 1.1 Découpage en Sprints

La caractéristique de la méthode agile la plus centrale à notre démarche est le découpage du projet en sprints: des sortes de sous projets visant chacun à implémenter une partie complète et fonctionnelle de bout en bout du produit final.

Dans notre cas, le projet était de base découpé en quatre incréments :

- Le langage "Hello World"
- Le langage "Sans Objets"
- le langage "Essentiel"

- Le langage “Complet avec l’extension”
- Le langage “Hello World” concerne uniquement quelques fonctionnalités basiques nécessaires pour que le compilateur puisse prendre une chaîne de caractères combinée avec une fonction “print” et la transformer en code assembleur qui permet d’afficher la chaîne de caractère en sortie.
- Le langage “Sans Objets” nécessite beaucoup plus de travail que le langage “hello world” et c’était le premier produit qu’il fallait rendre, ainsi il devait être intégralement développé et validé. Les fonctionnalités sont vastes et couvrent un large spectre d’un langage de programmation comme la déclaration de variables, les opérations arithmétiques et booléenne, les structures de contrôles et la manipulation des littéraux. Ce langage constitue une base pour une potentielle amélioration du compilateur d’où l’importance des choix de conception faits dans cette partie.
- Le langage “Essentiel“ représentait une phase intermédiaire du rendu final, l’objectif du sprint était de développer un langage avec lequel on puisse définir des classes, et la possibilité de leur associer des attributs et des méthodes. L’objectif était d’avoir un produit fonctionnel minimaliste pour pouvoir développer les tableaux et la bibliothèque du calcul matriciel.
- Le langage “Complet avec l’extension” était le dernier sprint réalisé dont l’objectif était d’implémenter les tableaux, et une bibliothèque de calcul matriciel et les fonctionnalités qui reste dans le langage avec objet comme le cast et instanceof. Une large partie du sprint a également été consacrée à faire des tests pour finaliser le produit et tout débugger.

## 1.2 Outils spécifiques utilisés

Pour mettre en oeuvre la méthodologie scrum, nous avons utilisé majoritairement deux fonctionnalités via l’outil Jira (équivalent de Trello):

- Le diagramme de Gantt
- Et le tableau de progression des tâches

Le diagramme de Gantt nous a permis de mieux visualiser le déroulement des différentes briques du projets sur toute sa longueur, de mieux les planifier et de mieux estimer leurs durées respectives. Il nous à également aidé à visualiser les interdépendances des tâches et les contraintes organisationnelles qui en découlent au travers notamment du concept de chemin critique, ce qui permet de plus clairement entrevoir les risques de retards. Le tableau de progression des tâches nous à quant à lui apporté une interface graphique pratique pour à la fois visualiser rapidement la progression des différentes user stories et des tâches qui en découlent. La possibilité de voir à quelle personne chaque tâche est attribuée était un plus pour cette fonctionnalité.

## 1.3 Dérivation méthodologique

Il est important de noter que nous n’avons pas exactement suivi la méthodologie scrum mais plutôt une méthodologie hybride. Cette différence s’est manifestée de plusieurs manières, qu’elles soient intentionnelles où qu’elles se soient imposées d’elles même pour faciliter l’organisation.

Tout d’abord, dans le cadre de ce projet, les user stories définies pour chaque partie ne sont pas des subdivisions d’un sprint indépendamment et parallèlement fonctionnels tel que pourrait l’exiger un utilisateur. Ce sont plutôt plusieurs étapes qui, une fois rassemblées, permettent de faire marcher l’intégralité du langage concerné par le sprint. Elles sont donc plus abstraites et techniques que si on avait choisis comme user stories les différents morceaux d’un langage (ex: us1: ”je veux pouvoir utiliser une instruction if”, us2: ”je veux pouvoir utiliser une instruction while” ect. . . ).

On peut désigner l’organisation en sprint comme “verticale”, quatre morceaux du rendu total qui implémentent chacun un langage du début à la fin en couvrant chacun toutes les étapes de la programmation pour

une partie du langage donnée. On peut désigner l'organisation dans les sprint comme horizontale, ou chaque étape de la programmation est réalisée l'une après l'autre. Cette méthode à permis à chacun de se spécialiser sur un bout particulier du code, on rentabilise ainsi le temps passer à comprendre une partie du projet avant de commencer à travailler. Cette méthode est favorisée par le fait que les étapes sont les mêmes pour chaque sprint.

Cet agencement à cependant mené à une autre divergence avec les principes du scrum, en effet la spécialisation des membres de l'équipe nous a amené à commencer chaque sprint avant la fin du précédent pour encaisser les problèmes de retard sur le planning.

La dernière divergence est l'ajout d'un quatrième sprint vers la fin du projet, regroupant l'extension et le langage complet, qui n'était à l'origine pas associés à un sprint particulier.

# 2 Organisation du groupe

Initialement, lors de la phase d'appropriation du projet, il n'y avait pas d'organisation prédéfinie. Après environ 3 jours, nous avons défini des rôles plus précis pour l'équipe. Mehdi est devenu le chef de projet, Victor le scrum master, Steven le responsable de l'extension et Jackson le responsable des tests.

Il y a également eu une répartition des différentes parties du projet (partie A, B et C) qui va de paire avec la méthode de travail décrite dans l'avant dernier paragraphe de la partie précédente. Le groupe s'est donc partagé en deux:

- Steven, Mohammed et Jackson se sont concentrés sur les parties A et B.
- Mehdi et Victor ont eux travaillés sur la partie C.

De même, certaines méthodes devaient être définies dans un grand nombre de classes (exemple: la méthode Verify à été implémentée pour de multiples classes différentes), auquel cas la tâche était assignée aux mêmes personnes qui les faisaient intégralement.

# 3 Découpage temporel du projet

Comme expliqué dans la partie 1.3, les user stories utilisées ici sont assez techniques. Elles se répètent généralement d'un sprint à l'autre à quelques exceptions prêt.
Les sprints sont eux mêmes découpés en groupes d'user stories, les étapes A, B et C. Elles correspondent à des ensembles cohérents du point de vue de la théorie de la compilation.
Ces étapes sont successives car chacune à besoin de la précédente pour marcher complètement. Cela ne nous a cependant pas empêcher de les faire se chevaucher temporellement pour pouvoir privilégier une bonne répartition du travail.
Dans l'étape A, on retrouve en premier la partie "décomposition de fichier deca en token", c'est à dire le découpage d'un programme en unités lexicales (en "mots") qui permettent ensuite de l'analyser grâce à la "construction de l'arbre lexicale". Cette analyse à pour but de comprendre le code mais aussi de détecter les erreurs syntaxiques que l'on gèrera dans une troisième parti, "gestion des erreurs syntaxiques".
Dans l'étape B, on a deux parties, "vérification contextuelle de l'arbre abstrait" et "décoration de l'arbre abstrait", C'est les parties où on s'assure de la cohérence des "mots" entre eux du point de vue des règles contextuelles du langage.
Finalement, dans l'étape C, on génère à partir de l'arbre abstrait le code assembleur correspondant.

## 3.1 Diagramme de Gantt prévisionnel

On peut voire dans les graphiques suivants que même si notre organisation d'équipe nous à mené à paralléliser beaucoup de tâches, on reconnait la configuration en escalier habituelle dans un diagramme de gantt.

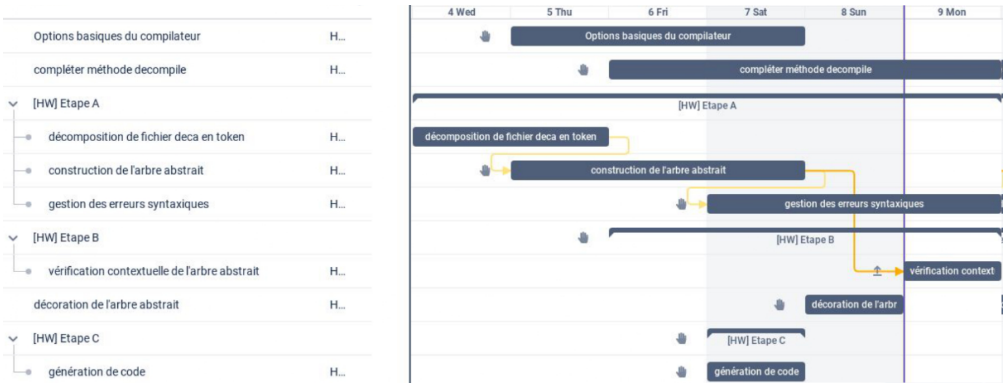


Figure 1: Gantt prévisionnel du sprint 1, langage "Hello World"

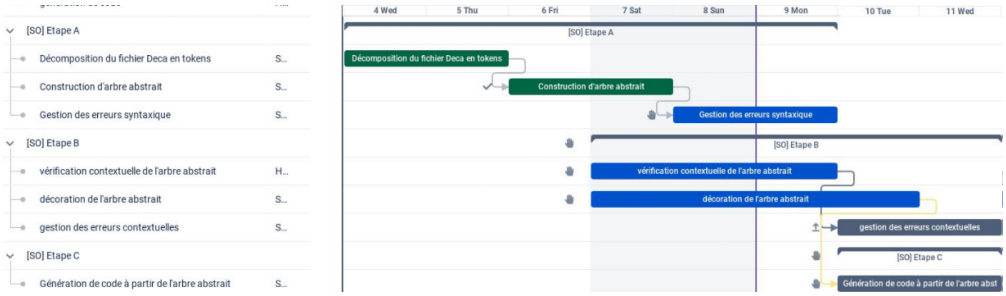


Figure 2: Gantt prévisionnel du sprint 2, langage "Sans Objets"

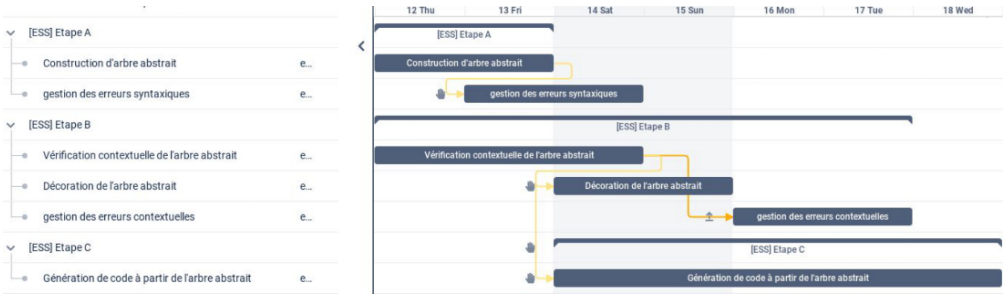


Figure 3: Gantt prévisionnel du sprint 3, langage "essentiel"

## 3.2 Diagramme de Gantt réalisé

On remarque ici comme dans la section précédente qu'une configuration en escalier respectant les liens de dépendances entre les user stories est visible. Le Gantt réalisé du sprint 1 (figure 4) à suivie exactement le prévisionnel, c'est parceque le langage "Hello World" était de loin le plus simple à implémenter, nous n'avons donc pas rencontré de difficultés majeurs à l'origine de retards. Ça n'a pas été le cas pour les


Figure 4: Gantt réalisé du sprint 1, langage "Hello World"

sprint 2 et 3 dont les langages, plus complexe, ont d'abords nécessité plus de temps de conception (en particulier le langage essentiel) puis ont été l'objets de plus d'erreurs et bugs qui ont pues se révéler très chronophages.


Figure 5: Gantt réalisé du sprint 2, langage "Sans Objets"

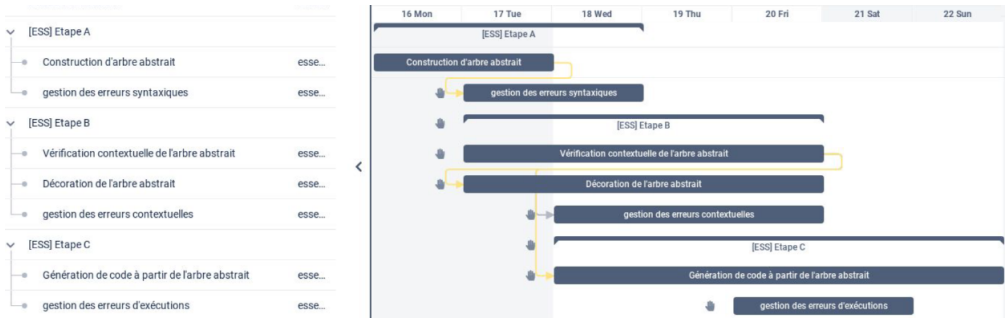


Figure 6: Gantt réalisé du sprint 3, langage ”Essentiel”

Le quatrième sprint (si-dessous) n’avait pas été prévu à l’origine. En effet, nous pensions travailler sur l’extension majoritèrement à cette période pour profiter de l’expérience accumulée lors du reste du projet, mais nous n’avions pas décidés d’en faire un sprint à part entière. Il inclut donc la majorité du travail sur l’extension (qui avait été entammé lors du sprint précédent), ainsi que le travail sur les deux élements manquants pour obtenir le langage complet.


Figure 7: Gantt réalisé du sprint 4, langage ”complet” et de l’extension

# 4 Obstacles majeurs rencontrés

## 4.1 Communication

- La communication n'était pas efficace au sein de l'équipe malgré les règles inscrites dans la charte.
- En conséquence, il est arrivé que deux personnes travail indépendamment sur le même morceaux de code ou que des personnes implémentent des fonctionnalités de leur propre chef fonctionnalités qui se sont révélés inutiles.
- Les outils de communications dont les rôles avaient été clairement définis se sont retrouvés fréquemment mélangés. Ces problèmes de communication ont occasionnés des pertes de temps parfois considérables. Restrospectivement, nous nous sommes rendus compte de l'importance de la rigueur d'une communication régulière et plus rigoureuse.

## 4.2 Répartition des taches

- L'interdépendance des parties du code faisait souvent qu'un groupe se retrovait bloqué en en atttendant un autre ayant prit du retard. Une meilleure attribution des tâches prennant en compte cette dépendance pourait être une solution.
- La différence de niveau et d'implication dans l'équipe rendait la tâche de répartition équitable du travail compliquée et souvent inefficace.
- Nos deadlines auto imposées prédatais souvent le rendu réel d'un jour ou plus, mais vers la fin elles étaient dépassées et le nos rendu se faisaient le dernier jour. Elles auraient dûes être mieux anticipées.

## 4.3 Respect de la charte

- Globalement, toutes les règles de la charte n'ont pas été respectées, notamment les réunions matinales qui n'ont pas été maintenues très longtemps après le début du projet puisque des réunions sans horaires fixes en cas de sujets importants à discuter leur ont été préférés.

# 5 Conclusion

Cette introduction à la méthode agile était une bonne expérience dans l'ensemble. Elle nous à aider à obtenir une meilleure apréhension des durées impliquées dans le processus de programmation ainsi que des diverses problématiques qui les causent et en découlent. L'exercice était rendu plus compliqué par la difficulté de prévoire la quantité de travail en jour que telle partie prendrait pour un projet que nous étions toujours en train de découvrir. La plupart du temps, cette quantité à été sous-estimée et nos propres deadlines dépassées. Mais c'est aussi ce qui c'est révélé formateur, car tout ces aléas nous ont forcés à nous adaptés en cherchant en permance à comprendre nos erreurs.