Projet Génie Logiciel

Bilan de Gestion d'Équipe et de Projet

Aurélien VILMINOT
Damien CLAUZON
Guilherme KLEIN
Léon ROUSSEL
Pierre ARVY

25 janvier 2022

Résumé

Ces quelques pages résument et critiquent l'organisation adoptée par notre équipe pour le projet de Génie Logiciel. On y retrouve quelques explications sur notre gestion d'équipe, une présentation de l'historique du projet, le temps passé sur les différentes activités (analyse, conception, développement, validation et documentation), ainsi que nos impressions personnelles. Ce bilan repose sur notre charte d'équipe et sur les plannings de prévision et effectif que le lecteur pourra trouver dans le répertoire `planning/` de notre projet sur *Git* (`Planning.pdf` et `Realisation.pdf`).

Table des matières

1 Gestion d'équipe

1.1 Objectifs

Le principal objectif de la gestion de projet et d'équipe était de mettre en place une organisation d'équipe efficace, et qui intègre tous les membres du groupe. Il était demandé de mettre en place un planning prévisionnel et de suivre l'avancement de notre projet, pour mettre en place des actions correctrices si un écart se creusait entre ceux-ci.

1.2 Travail d'équipe

Pour le développement du projet, il a été choisi d'utiliser la méthode agile *SCRUM*. Dans la charte d'équipe, nous nous étions autorisés à tourner les rôles à la fin de chaque sprint. Finalement, cela n'a pas été le cas. Chacun s'est approprié son rôle pour l'apprendre et le développer jusqu'au bout du projet. L'équipe a ainsi été composée de deux développeurs, deux testeurs et un *SCRUM-master* (qui aidait souvent les testeurs).

La première journée du projet a été réservée pour organiser le premier sprint et découper l'ensemble du projet en fonctions "client", dites *user stories*, que nous avons ensuite écrites sur Trello. Nous y avons associé une certaine quantité de travail avec un *planning poker*. Les suivants ont été faits à l'oral lors de la réunion hebdomadaire du lundi (organisant le prochain sprint comme spécifié sur la charte). La plupart des tâches composant les sprints avaient été écrites dans le planning prévisionnel `Planning.pdf`. Finalement, comme il est possible de le voir sur le planning effectif `Réalisation.pdf`, la plupart ont changé en cours de route. Il a fallut quelques jours à l'équipe pour adopter un esprit de développement "agile".

Comme spécifié sur la charte d'équipe, un *stand-up meeting* a été tenu chaque jour. Ces réunions ont permis à chacun de s'attribuer l'avancement du projet, et de partager ses idées sur la suite de celui-ci. La tenue du cahier de bord, résumant ces réunions, a été très utile pour construire un historique du projet.

Enfin, concernant le travail collaboratif, l'utilisation de l'outil de versionnage *Git* a été très utile. La séparation complète du développement et des tests s'est montrée très efficace. Après une prise en main difficile la première semaine (trop de branches, conflits...), chacun pouvait travailler de son côté en récupérant le travail des autres membres du groupe.

2 Historique du projet

2.1 Division des tâches

2.1.1 Échelle globale

Comme le suggère le planning prévisionnel de notre équipe, l'ensemble du projet a été divisé en trois sprints de cinq jours. Comme chacun devait apporter une nouvelle fonctionnalité utilisateur, il a été choisi d'implémenter, valider et documenter chaque partie du polycopié en un sprint. En plus de celles-ci, l'extension et l'analyse énergétique ont été étalées tout au long du projet. Ainsi, comme le suggère la figure 1, le premier sprint devait apporter la partie fonctionnelle de "Hello World" (conception et validation), la documentation associée et les résultats de recherches bibliographiques pour l'extension et l'analyse énergétique.

2.1.2 Échelle du sprint

Sur la figure 1, on retrouve également la division des parties conception et validation pour chaque sprint. Avant de commencer à écrire du code, une analyse commune était faite (souvent en échangeant après les *stand-up meeting*). Une fois que l'équipe s'était mise d'accord, l'implémentation pouvait commencer. Pour le premier sprint, chaque partie commencée devait être terminée avant de passer à la suivante. Pour les deux suivants, et surtout pour le dernier, pouvant être considéré comme un *epic sprint*, chaque partie été avancée pour livrer une fonctionnalité du produit (en se référant au *product backlog* de Trello), avant d'être reprise pour ajouter la suivante. Par exemple, lors du sprint 3, l'accès aux attributs d'une classe a été implémentée, avant de revenir sur chacune des partie pour proposer

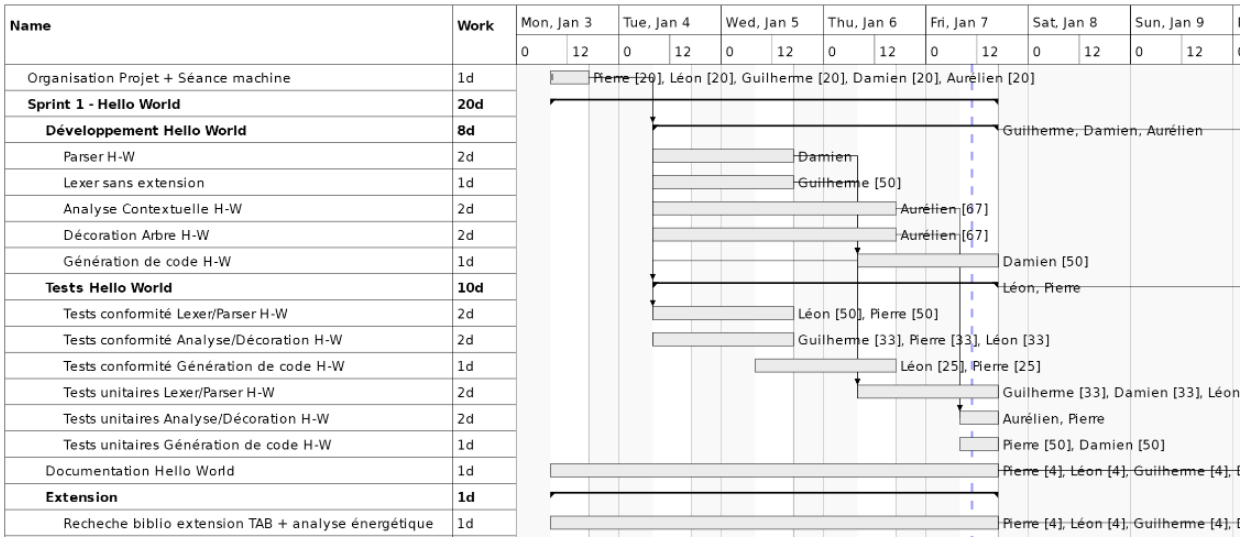

FIGURE 1 – Capture d'écran du planning prévisionnel de notre équipe. On y retrouve la division des différentes tâches composant le sprint 1.

ensuite la bonne visibilité de ceux-ci dans une hiérarchie de classes.

De manière assez classique, le développement du *Lexer*, du *Parser*, de l'*Analyse Contextuelle* ainsi que de la *Génération de Code* s'est fait de manière distincte. A chaque fois, le *Parser* et l'*Analyse Contextuelle* ont été démarrés en parallèle. Cette dernière demandant plus de temps, la *Génération de Code* démarrait dès que le *Parser* était fini, et été livrée seulement une fois l'*Analyse Contextuelle* finie. Il est à noter que le *Lexer* complet a finalement été livré lors du premier sprint.

L'étape de validation a été la même pour chaque sprint. Les tests de conformité (valides et invalides) de chaque partie incluent des tests "boîte-noire" et "système" (avec la technique "oracle" pour ceux-ci). Ils devaient être écrits le plus rapidement possibles de manière à ce que les développeurs détectent rapidement leurs erreurs et les corrigent en conséquence. Pour les deux premiers sprints, les tests "unitaires" étaient écrits conjointement par les développeurs et testeurs après le développement des parties concernées. Pour le sprint 3, ceux-ci ont été écrits avant le développement pour se rapprocher d'une vraie méthode de développement dirigé par les tests. Contrairement à ce qui est annoncé dans le planning prévisionnel, seule l'analyse contextuelle demandait des tests unitaires.

Enfin la documentation de chaque partie a été écrite le plus rapidement possible après son implémentation, et avant la fin de chaque sprint. Souvent, le vendredi après-midi était réservé à l'écriture de celle-ci.

2.2 Temps effectif

Cette partie résume le temps passé sur les principales parties du projet que sont l'analyse, le développement (conception et codage), la validation et la documentation. On y retrouve aussi les dates clés du projet. Les trois figures suivantes présentent le nombre d'heures passées sur chaque partie, et composant chaque sprint. Le total d'heures sur chaque sprint oscille entre 150h et 160h. Il est à noter que certaines tâches ne sont pas résumées dans les figures suivantes, les ayant considérées comme minimes ou non pertinentes. Ainsi, l'automatisation des tests, la préparation des suivis, ou encore la gestion de l'environnement de travail ne sont pas considérées.

2.2.1 Analyse et Développement

Lors du premier sprint, le développement n'a pas démarré à la vitesse initialement prévue. En effet, beaucoup de temps a été passé sur l'analyse du polycopié. L'analyse du second sprint a également été importante pour s'approprier la génération de code (qui était presque inexistante pour le sprint 1). Les dates suivantes résument les livraisons

fonctionnelles du projet :

6 Janvier 2022 (sprint 1) : Fin partie "Hello World".

11 Janvier 2022 (sprint 2) : Déclaration de variable et assignation (entier, flottant, booléen).

13 Janvier 2022 (sprint 2) : Boucle `while`, conditionnels (`if/else`).

18 Janvier 2022 (sprint 3) : Déclaration et assignation de classes, d'attributs.

20 Janvier 2022 (sprint 3) : Visibilité des attributs, déclaration et appel de méthodes.

21 Janvier 2022 (sprint 3) : Héritage, retour de méthodes.

Développement Compilateur & Analyse	Sprint 1	Sprint 2	Sprint 3	Total
Lexer	5	0	0	**5**
Parser	12	7	4	**23**
Analyse Contextuelle	15,5	9	12	**36,5**
Génération de code	3	27	35	**65**
Compilateur	5	4	1	**10**
Total Développement	**40,5**	**47**	**52**	**139,5**
Total Analyse	**30**	**28**	**15**	**73**

FIGURE 2 – Temps passé (en heures) à analyser les différents problèmes et à développer le compilateur *decac*.

2.2.2 Validation et Documentation

Le temps passé à valider le développement a à peu près été le même pour chaque sprint. Quelques jours après le début de projet, les testeurs ont établi un plan de validation à suivre (voir la documentation *Validation*). Celui-ci a permis de gagner en efficacité et du temps pour les sprints 2 et 3. Les dates suivantes résument les moments clés de la validation du compilateur :

11 Janvier 2022 (sprint 2) : Rédaction du plan de validation.

13 Janvier 2022 (sprint 2) : Fin de validation du code relatif au sprint 2.

21 Janvier 2022 (sprint 3) : Fin de validation du code relatif au sprint 3.

22 Janvier 2022 (sprint 3) : Fin de validation du code relatif à l'extension.

Validation Compilateur & Documentation	Sprint 1	Sprint 2	Sprint 3	Total
Conformité Lexer/Parser	25	15	8	**48**
Conformité Analyse Contextuelle	12	15,5	12	**39,5**
Conformité Génération de Code	5	13	12	**30**
Unitaires Analyse Contextuelle	17	17	16	**50**
Total Validation	**59**	**60,5**	**48**	**167,5**
Total Documentation	**15**	**11**	**15**	**41**

FIGURE 3 – Temps passé (en heures) à écrire des tests pour valider le compilateur et à écrire la documentation.

2.2.3 Extension et Analyse énergétique

Les recherches bibliographiques pour l'extension et l'analyse énergétique ont commencé dès le début du projet. Lors du deuxième sprint, la grammaire Deca a été étendue pour y ajouter les éléments relatifs à l'extension TAB, livrée le dernier jour du sprint 3. Les dates suivantes résument les moments clés de l'extension et de l'analyse énergétique :

14 Janvier 2022 (sprint 2) : Fin de l'extension de la grammaire Deca pour y ajouter les tableaux.

21 Janvier 2022 (sprint 3) : Fin de l'analyse énergétique.

22 Janvier 2022 (sprint 3) : Livraison fonctionnelle de l'extension.

Extension et Analyse énergétique	Sprint 1	Sprint 2	Sprint 3	Total
Recherches Bibliographiques	8	5	0	**13**
Extension de la grammaire	0	6	2	**8**
Développement Parser/Analyse Contextuelle	0	0	6	**6**
Génération de Code	0	0	12	**12**
Tests/Bibliothèque Calcul Matriciel	0	0	10	**39**
Total Extension	**4**	**0**	**30**	**65**
Total Analyse Energétique	**4**	**5**	**8**	**17**

FIGURE 4 – Temps passé (en heures) à travailler sur l'extension (analyse, développement et validation) et sur l'analyse énergétique.

3 Critique

Sur l'ensemble du projet, différents aspects peuvent être abordés afin d'en faire une critique constructive, ayant pour but d'améliorer les futurs projets que nous réaliserons.

3.1 Les points positifs

Les *stand-up meetings* se sont montrés très efficaces sur l'ensemble du projet. Ces derniers nous ont permis d'échanger efficacement sur les tâches réalisées la veille et qui seront effectuées pendant la journée. Ainsi, chacun peut organiser sa journée en fonction des besoins des autres. Ils permettent également à l'ensemble de l'équipe de prendre du recul sur l'avancement global du projet. De plus, le fait que le *scrum-master* demande à chacun s'il a ou non des éléments à faire remonter a facilité les échanges au sein du groupe.

La cohésion d'équipe était de mise tout au long du projet. Nous nous connaissions auparavant et cela nous a permis d'avoir une équipe soudée dès le départ. Des activités extra-scolaires ont été organisées chaque semaine afin de maintenir, voir renforcer, cette cohésion (apéritifs dînatoires, fête de l'Épiphanie…).

L'établissement d'une charte d'équipe stricte au début du projet a permis de fixer des règles nécessaires au bon déroulement du projet. Nous pouvons notamment citer les sanctions établies pour les retards, qui ont été respectées et suivies par les retardataires (gainage, viennoiseries). De plus, les contraintes d'emploi du temps ont permis d'assurer la communication au sein du groupe. S'imposer des règles et des objectifs a développé une forte motivation au sein de l'équipe. Notre objectif : *" réaliser le meilleur compilateur de l'Ensimag"*.

La division du travail mise en place a engendré une très bonne coordination de l'équipe durant le projet. Les développeurs ont pu aider les testeurs, et inversement. Tout le monde a ainsi pu explorer chacun des postes de façon plus ou moins directe.

Enfin, le dernier point positif du projet concerne les méthodes de gestion de projet. Nous avons pu en expérimenter certaines. À l'issue de ce projet, en prenant du recul, nous nous sommes rendus compte de l'importance et de l'efficacité de ces méthodes. Cela concerne notamment la parallélisation des tâches ou encore l'organisation des différentes réunions qui peuvent, au début, paraître être une perte de temps.

3.2 Les axes d'amélioration

Néanmoins, malgré de nombreux points positifs, des axes d'améliorations sont envisageables. Premièrement, il a été admis par tous les membres que faire reposer certaines parties cruciales sur une seule personne était une erreur. En effet, si cette dernière ne peut plus être disponible(maladie…), alors le projet pourrait être en péril. Cette erreur a été commise à deux reprises : l' *Analyse contextuelle* reposait uniquement sur Aurélien tandis que la *Génération de code* était gérée exclusivement par Damien. Bien que les testeurs avaient un certain recul sur la partie concernée, ils n'avaient pas la capacité de reprendre rapidement le rôle de développeur.

Bien qu'il était recommandé par les enseignants de commencer l'extension dès le début du projet, nous avons fait le choix de la commencer à partir de la deuxième semaine. L'analyse et le début de conception ont été réalisés par les testeurs. Ces derniers ont donc dû mettre temporairement de côté l'écriture de tests car le temps restant était trop court pour que les développeurs puissent assurer à la fois le langage avec-objet et l'extension.

Nous avons rapidement compris l'importance des tests et l'intérêt d'obtenir une couverture de tests élevée. Le rôle de testeur ayant été défini dans la charte, ces derniers ont rapidement pu commencer à fournir des jeux de tests. En revanche, cette validation effective s'est réalisée avec précipitation, ce qui les a conduit à établir un plan de validation.

Enfin, l'ensemble des outils proposés par *GitLab* n'ont pas été exploités pour la partie de gestion de projet. Le *Burndown chart* et la planification auraient notamment pu être réalisés automatiquement par *GitLab* afin d'avoir une meilleure intégration du projet.

4 Bilans personnels

4.1 Aurélien VILMINOT

«Ce projet a été une opportunité de mettre en application les connaissances assimilées pendant mon cursus. Il m'a permis de me rendre compte de l'importance des matières transversales telles que la gestion de projet qui peuvent nous apparaître comme secondaire en première année.

J'ai ainsi pu découvrir de nouveaux outils comme *Maven* et comprendre comment fonctionne un compilateur en interne au sein d'un ordinateur. De plus, l'extension choisie nous a donné une totale liberté de sa conception à son implémentation. Habituellement, la partie de conception est déjà faite dans les sujets fournis mais dans ce cas présent, j'ai pu prendre conscience de la difficulté à établir des spécifications précises. »

4.2 Damien CLAUZON

«Dédier tout son temps sur un projet de si grande ampleur a été une expérience très enrichissante pour moi. J'ai beaucoup appris pendant ce mois passé en équipe, notamment en terme de gestion de projet en s'essayant aux méthodes agiles.

Outre les compétences acquises et les outils découverts, j'en retiens surtout les nombreuses problématiques introduites par le travail en équipe auxquels il a fallu faire face tout au long du projet. »

4.3 Guilherme KLEIN

«C'était une excellente occasion de jouer un rôle dans une équipe au-delà du développement. J'ai pu voir en pratique la façon de travailler avec des tests et des méthodes agiles et ainsi e apprendre beaucoup sur pour s'organiser au mieux en équipe projet.

La partie compilateur m'a servi à relier des sujets très variés que j'ai étudiés à l'université. Je vais certainement utiliser ce que j'ai appris de ce projet dans mes futurs travaux. »

4.4 Léon ROUSSEL

«Je pense que ce projet a été l'occasion de commencer à trouver des méthodes de travail qui vont servir dès la sortie de l'école. L'ampleur du projet et le nombre conséquent de membre par équipe (5) nous a forcé à développer des compétences de gestion de projet, sans lesquelles nous n'aurions pas pu avancer aussi sereinement.

Pour ma part, ce fut un mois consacré plus particulièrement aux tests des fonctionnalités du compilateur. Bien que la création de tests s'avère être une tâche assez répétitive, c'est un domaine dans lequel il faut travailler avec méthode et précision. Au-delà des tests, cette période de projet m'a permis d'acquérir des compétences techniques indispensables, ce qui n'est pas forcément le cas dans la filière MMIS. Cela permet donc de se diversifier.

L'exercice qui nous est proposé est très intéressant car il permet de mettre en oeuvre une grande partie des connaissances acquises à l'Ensimag, de l'assembleur à la théorie des langages. Je trouve que le survol de toutes ces notions en un peu plus de trois semaines donne beaucoup de clés pour la suite. »

4.5 Pierre ARVY

«Ce projet aura été l'occasion d'appliquer toutes les compétences techniques et transversales apprises depuis mon arrivée à l'Ensimag. Le développement et l'écriture des tests ont fait appel à tous les réflexes développés dans les cours de programmation suivis depuis lors. Ce projet amène également beaucoup de compréhension sur l'étape de compilation de certains langage de programmation.

Enfin, et surtout, la mise en place d'une organisation d'équipe et d'une réelle collaboration a permis de s'immerger dans un cadre réel d'entreprise. J'en retiens l'importance d'une bonne communication entre les membres d'une équipe, et la force que peut avoir une méthode agile sur un projet de génie logiciel. L'organisation, pouvant paraître fastidieuse a priori, se révèle finalement cruciale pour s'assurer du bon déroulement d'un projet. »

5 Bilan global

D'un point de vue technique, le projet de Génie Logiciel nous a permis d'apprendre à utiliser de nouveaux outils d'aide au développement, comme *Maven*, *Git* ou encore l'IDE *Intellij IDEA*. Les six aspects citées dans la partie [Motivation] du polycopié ont été expérimentés durant le projet. Celui-ci apporte une véritable plus-value dans la formation d'ingénieur Ensimag. Il nous aura permis de nous confronter aux véritables problématiques qu'un ingénieur logiciel peut rencontrer en entreprise.

Chacun des membres de l'équipe en ressort grandi humainement, et professionnellement. Être focalisé sur un unique projet pendant un mois permet de s'immerger dans un cadre réel d'entreprise. Ainsi, nous avons pu prendre conscience de l'importance de mettre en place une organisation d'équipe efficace et intégrant tous les membres de l'équipe. Elle permet d'améliorer la productivité globale de l'équipe, ainsi que de limiter aux maximum les conflits au sein de celle-ci.