Compte-rendu Release#1- groupe 5

Objectifs

Pour cette première release, nous avons défini que l'objectif était d'avoir un logiciel qui non seulement interprète le MiniJaja mais aussi le compile en JajaCode et ainsi interprète le JajaCode. Pour cela nous avons défini 3 objectifs principaux et quelques objectifs secondaires à completer en 3 sprints :

Principaux :

1. Créer le Parser/Lexer pour un sous-ensemble du MiniJaja correspondant aux opérations arithmétiques et comparateurs booléen (Avec JavaCC).
2. Créer l'AST (avec JavaCC) et l'interpréteur de ce sous-ensemble du langage.
3. Compiler en JajaCode le sous-ensemble correspondant aux parties précédentes et interpréter ce JajaCode.

Secondaires :

- Avoir une interface graphique qui permet d'éditer du code, importer/sauvegarder des fichiers
- Pourvoir exécuter le code depuis l'interface graphique
- Avoir un contrôleur de type fonctionnel
- Implémentation de la mémoire (pile/tas)

Pour ce qui est des objectifs principaux, nous avons pu compléter les deux premiers dans les temps impartis (lors des deux premiers sprints). Pour ce qui est du troisième, il est majoritairement fini mais quelques fonctionnalités sont encore manquantes sur l'interpréteur Jajacode.

Stratégies

Afin de mettre en oeuvre ce projet, nous avons utilisé plusieurs approches agiles.
Tout d'abord, afin d'organiser l'équipe nous avons décidé de choisir un Scrum Master ainsi qu'un responsable de projet. Le responsable de projet aura pour tâche principale de centraliser et diffuser les informations ainsi que de s'assurer de l'évolution continue de l'application. Quant au Scrum Master, il aura pour but de s'assurer des bonnes pratiques de l'équipe par rapport aux méthodes agiles.

Pour la rédaction des User Stories, nous nous sommes réunis afin de discuter des multiples possibilités qui doivent être disponible au moment de la release#1. Pour cela, nous avons listé plusieurs scénarios et les fonctionnalités qui étaient nécessaires à la réalisation de ces derniers.
Par la suite, nous avons découpés ces User Stories en tâches réalisables par une personne. Ces tâches ont vu leur complexité évaluée ainsi que leur priorité selon les objectifs du sprint, les besoins de l'application mais aussi selon les besoins d'implémentations (algorithmes,...). Par exemple, au sprint 1,

bien que les tâches d'interprétations du Jajacode aient une complexité élevée étant donné que cela implique l'implémentation de visiteurs pour l'AST, leur priorité était faible au vu des objectifs du sprint et de l'état de l'application.

Les tests représentant une partie très importante du développement nous avons désigné une personne pour le rôle de Tester qui a pour but de réaliser plusieurs types de tests : tests d'intégrations, paramétriques,… En revanche les tests unitaires correspondant à une ou plusieurs classes sont réalisés par l'ensemble de l'équipe. De plus, les scénarios de tests correspondant aux user stories permettent d'avoir une direction claire sur l'écriture des tests et le développement de l'application.

Pour finir sur les revues, sous la responsabilité du responsable de projet nous avons effectué à chaque fin de sprint une revue afin de discuter des tâches effectuées, des tâches restantes, des méthodes de travail, etc. Nous effectuons aussi sur chaque demande de Merge, une revue du code et des tests, ce qui permet de garder un œil sur l'avancement de l'application mais aussi sur la qualité du développement et des tests.

CI/CD

Pour la CI/CD, aussi appelée intégration et développement continu, nous utilisons les jobs gitlab. On utilise un fichier qui définit l'entièreté des jobs à exécuter lors d'un merge ce qui nous permet d'avoir un œil sur la compilation de l'application et l'exécution des tests et ainsi avoir une branche toujours fonctionnelle.

Communications

Afin de communiquer au sein du groupe nous utilisons la majeure partie du temps Discord, avec un serveur dédié, ce qui nous permet d'être réactif et au courant des dernières nouvelles concernant l'application. En plus de ce serveur, nous avons l'habitude de communiquer lors de séances de TP afin de résoudre certains problèmes ou bien pour le développement de certaines fonctionnalités.

Revues sprint

Chaque revue de sprint comporte plusieurs points. Notamment les tâches réalisés, les tâches à réaliser au prochain sprint et enfin les avis des différents développeurs par rapport au sprint.
Sprint 1 :

- Fait :
  - Transformation en LL(1) de l'ensemble de la grammaire MiniJaja
  - Mise en place de Javacc/JJtree
  - Parser/Lexer MiniJaja sur l'ensemble de la grammaire MiniJaja

    - Première version de la pile
    - Editeur de texte et terminal d'affichage sur l'interface graphique
    - Tests sur le parser/lexer
- A réaliser au prochain sprint :
    - Définir les nœuds de l'ast
    - Sauvegarder/importer fichier
    - Finir l'implémentation de la pile
    - Implémenter l'interpréteur MiniJaja sur le sous-ensemble défini
    - Implémenter CI/CD
- Discussions :

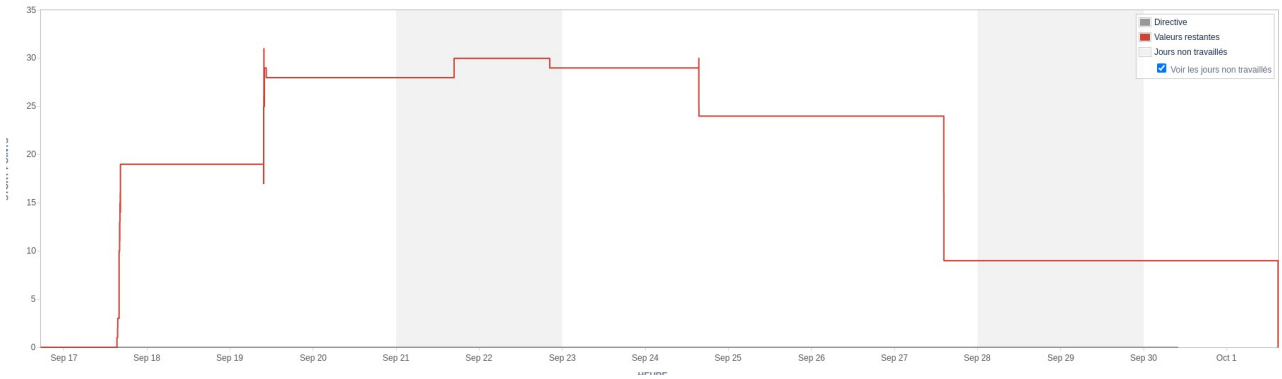

Figure 1: Burdown Chart Sprint 1

Sprint 2 :

- Fait :
    - Création de l'AST
    - Interpréteur du sous-ensemble de MiniJaja
    - Importer/Sauvegarder fichier
    - Implémentation de la table des symboles et de la pile (complet)
    - Début du contrôle des types
    - Exécution du programme sur la l'interface graphique
    - Test des exceptions
- A réaliser au prochain sprint :
    - Implémenter CI/CD
    - Finir contrôle des types
    - Implémentation tas
    - Parser/Lexer sur l'ensemble du JajaCode
    - Compilation JajaCode du sous-ensemble défini
    - Interprétation JajaCode du sous-ensemble défini
    - Ouvrir plusieurs fichiers sur l'éditeur de texte
    - Dark mode pour l'éditeur
    - Afficher résultat interprétation sur le terminal de l'éditeur
- Discussions :
    - Etre plus impliqué sur le travail
    - Travailler 1 ticket à la fois et pas celui des autres
    - Faire tourner le rôle de Scrum Master
    - Notifier du travail fini avec les tickets

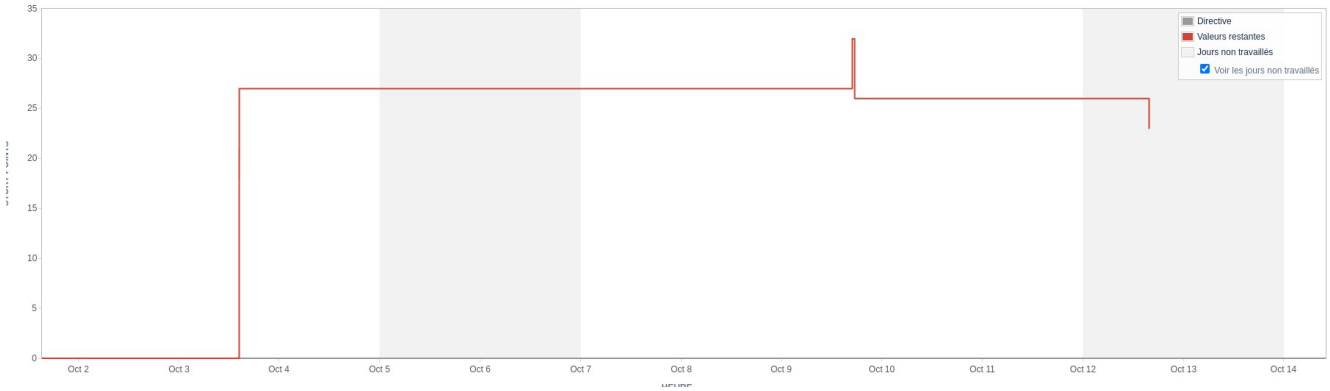

Figure 2: Burdown Chart Sprint 2
(A noter qu'une erreur de manipulation sur Jira, à la fin du sprint 2, fait que nos tickets validés ne sont validés qu'au début du sprint 3)

Sprint 3 : A venir (Apres rendu CR)

Tests

Analyse des pratiques

Pratiques fonctionnelles :

- Assigner un rôle de Tester
- Assigner un développeur sur l'interface graphique à chaque sprint
- Désigner un Scrum Master
- Communication par Discord et revue de sprint en personne.
- Désigner 2 développeurs à chaque merge request pour effectuer la revue de code

Pratiques non-fonctionnelles :

- Laisser la responsabilité de Scrum Master à une seule personne. En effet, cela ne permet pas de responsabiliser l'ensemble de l'équipe et donc rend les développeurs moins impliqués et moins conscients des méthodes agiles.

Indications pour la suite

Pour la suite nous avons plusieurs pistes d'améliorations :

- Changer la Scrum Master chaque semaine pour responsabiliser l'ensemble de l'équipe lors du développement. Jusqu'à maintenant le Scrum Master était la même personne.