Rapport Personnel Compilation : Alexandre Quettier

En tant que membre de l'équipe, je n'ai pas eu de tâche spécifique à réaliser. Je me suis davantage chargé d'avancer le travail avec mes collègues et donc d'avoir une vue d'ensemble du projet en participant à toutes les réalisations.

1er délivrable

- Dans le cas du *Pretty Printer* qui représente le premier délivrable, nous nous sommes mis en groupe dans l'unique but de saisir la logique et l'utilisation de *Xtend* et de *Xtext*. Nous avons ainsi pu comprendre le besoin de créer des attributs au sein de notre grammaire pour pouvoir les "pretty printer".
- Avec l'aide de Kevin Ledy, j'ai pu créer l'ensemble des attributs nécessaires dans notre grammaire. Nous avons réalisé cette tâche à deux pour trouver un nommage des attributs qui correspondrait à l'ensemble du groupe.
- Avec l'aide des autres membres du groupe, j'ai pu réaliser quelques éléments du pretty printer, notamment l'intégration au sein de tout les éléments de notre grammaire de la tabulation.
- En parallèle au *Pretty Printer*, j'ai pu réaliser quelques fonctions de test. C'est à des fins de test que j'ai réalisé un *Ugly Printer*. Le but de ce *Printer* est de placer tout le texte d'un fichier *WHILE* sur une seule ligne. Une fois cela fait, on passe le fichier dans le *Pretty Printer* et le résultat doit correspondre au fichier source pretty printé (sans modification du *Ugly Printer* donc).

2ème délivrable

- Une fois que le 1er délivrable fut livré au client, mes collègues se sont attelés à intégrer une table des symboles dans notre projet. Avec l'aide de Sébastien, nous nous sommes occupés quand à nous de réaliser des *stress tests* de notre *Pretty Printer* pour le pousser dans ses derniers retranchements. Il a donc fallu réaliser des tests en longueur (un fichier de plus en plus long), en largeur (un fichier avec des lignes de plus en plus longues) et enfin en profondeur (un fichier avec de plus en plus d'imbrications).
- J'ai également effectuer des tests sur un type d'objet de *Xtext* qui se nomme le

validator. Je voulais voir s'il était possible de réaliser une gestion des erreurs d'un fichier *WHILE* avant de lancer la compilation vers le *CPP*. Toutefois, ces recherches se sont révélés infructueuses et j'ai préféré faire confiance à mes collègues et à leur travail sur le *ThreeAddGenerator*.
- Une fois ces *stress tests* parfaitement fonctionnels, j'ai pu rejoindre mes collègues dans leur travail sur le générateur de code 3 adresses. En effet, pour simplifier la traduction vers du *CPP*, nous avions décidés de passer par du code 3 adresses. Ce générateur est donc une composante fondamentale et essentielle à notre compilateur.

3 ème délivrable

- Pour ce délivrable, je me suis fortement concentré sur la génération du code 3 adresses, notamment la correction de quadruplet incorrect (comme les *cons* ou les expressions booléennes). En effet, malgré le travail de Pierre-Henri Collin et de Kevin Ledy, il subsistait des erreurs qui étaient liés à la façon dont la grammaire était conçu. Il a donc fallu modifier la grammaire pour mieux générer ensuite le code 3 adresses nécessaire.
- En parallèle, j'ai réalisé l'interface de compilation en *Java*, nécessaire au lancement du compilateur en ligne de commande. Cette interface va avoir pour tâche de décomposer la ligne de commande qu'on lui fournit afin d'isoler les paramètres de compilation souhaités (nom du fichier de sortie, option de débuggage, etc).
- Enfin, durant les derniers jours avant la remise du projet au client, nous nous sommes aperçus que certains composants du langage *WHILE* n'était pas compilés en langage 3 adresses. C'était notamment le cas des expressions booléennes *AND, OR, =?* … Après le premier rendu au client, il a donc été vital de réaliser ces codes 3 adresses pour disposer d'un compilateur pleinement fonctionnel avant l'ultime rendu. Plusieurs jours ont donc été nécessaire à la fin pour réaliser cette génération de code.