Université de
Cergy-Pontoise

Rapport de projet

Projet Yolodt

Auteurs :
Julien Abadji
Bastien Lepesant

Jury :
M. Marc Lemaire
M. Tianxiao LIU

5 Janvier 2015

Table des matières

1 Introduction

1.1 Contexte

Nous sommes en Licence 2 Mathématiques-Informatique à l'Université de Cergy-Pontoise. Parmi les enseignements dispensés au cours du premier semestre se trouve un module de programmation orientée objet (POO), en Java. C'est dans ce contexte d'introduction à la POO que nous a été proposé le projet (baptisé Yolodt) dont il est question.

1.2 Objectif

Yolodt est un programme qui a pour objectif de proposer une indexation de fichiers de type "Open Office Document", d'un stockage de leur titres, puis d'une recherche ultérieure d'informations contenues dans ceux-ci.

1.3 Environnement

Le projet a été développé avec le Java Development Kit 7 (JDK 7), à l'aide de l'Environnement de Développement Integré (IDE) Eclipse Luna. Les systèmes d'exploitation utilisés durant toute la durée du développement ont été Debian, ArchLinux et ElementaryOS. Ces trois systèmes sont de type Linux.

1.4 Composition

Le groupe est composé de Julien Abadji et Bastien Lepesant. Nous avons des façons différentes de réfléchir, et nous avons perçu cette différence comme une force : Nos visions différentes permettent d'entrer dans une confrontation d'idées régulière, aboutissant à des argumentations bénéfiques pour la qualité du projet.

2 Conception et réalisation

2.1 Partir de l'utilisateur pour arriver au diagramme.

En premier lieu il nous a semblé intéressant de s'imaginer le comportement d'un utilisateur devant le programme fini. Ainsi, au fil des actions du personnage virtuel, l'allure globale du projet se détaille.

Le schéma d'utilisation classique part d'une importation d'un ensemble de dossiers ou fichiers de type Open Document Text (abrégé ODT par la suite), l'extraction d'un fichier contenant les données à analyser, l'analyse, et le stockage, ces trois dernières étapes étant totalement transparentes pour l'utilisateur. Ensuite vient la recherche. L'utilisateur doit pouvoir entrer un ou plusieurs mots et avoir en retour une liste triée par pertinence des documents ODT les plus pertinents.

L'étape suivante consiste à abstraire le problème en passant par un diagramme de classes UML. Après différentes discussions, le diagramme approuvé par notre groupe est celui-ci.

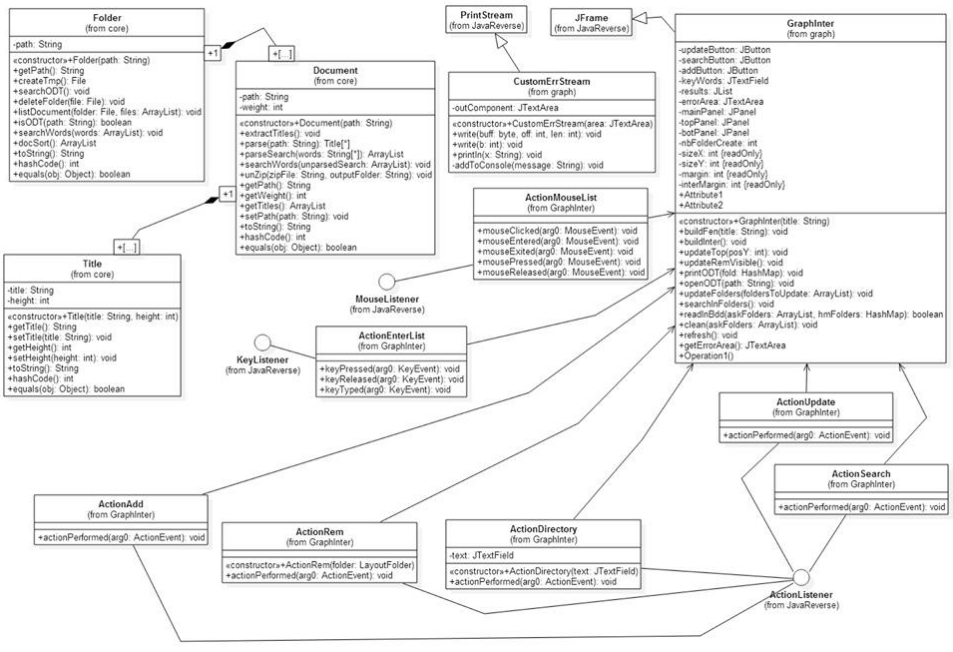

FIGURE 1 – Voir l'UML en plus grand

2.2 Du diagramme au code.

Au vu de la taille du projet, la réalisation a nécessité un partage des tâches. Afin d'avoir une gestion du projet efficace, simple et sûre, un dépôt git [1] a été créé. Ainsi, chaque modification du code est expliquée, enregistrée, et réversible en cas de problème.

Concernant le partage des tâches suscité, Bastien s'est plus occupé des aspects d'entrée/sortie, de gestion des fichiers et de l'interface graphique, tandis que Julien s'est focalisé sur l'extraction, l'analyse (parsage XML) et l'interface en ligne de commande.

Le partage n'a cependant pas été hermétique : Nous nous avous mutuellement relu notre code et suggéré des améliorations. Un exemple est la gestion de la recherche. Initalement réalisée par la méthode searchWords écrite par Bastien, Julien l'a modifié légèrement et y a adjoint la méthode parseSearch, afin de proposer des possibilités de recherche plus étendues.

1. git est un logiciel de gestion de versions.

3 Manuel utilisateur

Le programme est utilisable en deux versions : Une première non-interactive en ligne de commande, et une seconde interactive graphique.

3.1 Interface en ligne de commande

Le programme peut être invoqué avec 3 arguments différents. L'abscence de arguments renvoie une erreur et un rappel sur la syntaxe des arguments attendus.

-d <folder1> <folder2> ... : Ajoute les dossiers folder1 folder2 ... à la base de données, c'est-à-dire recherche récursivement tous les ODT présents dans ces dossiers, extrait et analyse leur contenu et enregistre les titres pour une recherche future.

-f <file> : Affiche les informations importantes de l'ODT file : Chemin, titres.

-w <words> : Retourne les ODT les plus pertinents en fonction des mots recherchés. Si un "+ " est envoyé, la recherche se fera en mode ET (ie. on cherchera un ODT contenant tous les mots spécifiés après le +). Si un ensemble de mots est spécifié entre guillemets, le programme ne retournera que des ODT contenant exactement la phrase entre guillemets.

Conclusion

Ce projet a été pour nous l'occasion de découvrir le travail collaboratif sur un projet concret. Nous avons constaté que cette méthode de travail avait des avantages indéniables (productivité, apports de l'argumentation et de la revue de code..), mais pouvait potentiellement être très chronophage et ralentissante sans planification des tâches et utilisation d'outils de travail collaboratifs (git). De plus, ce travail nous a permis de mesurer l'importance de la communication dans un projet à plusieurs : en effet, sans une quasi constante remise en question de nos idées le projet aurait pu être bien plus complexe à réaliser.

Durant nos tests nous avons constaté un bon fonctionnement du programme. Cependant, il est coutume de dire qu'un projet n'est jamais fini. De nombreuses améliorations sont à imaginer : Effectuer la recherche dans les paragraphes, améliorer l'interface non-graphique afin de l'adapter à une utilisation par un humain et par un script (avec des sorties en XML, JSON, etc.)...