Baptiste PAULETTO - Louis UNG - Fabien GELUS – Mengxin ZHANG
B3246 B3252

Conclusion Projet GL/UML 2019

A la suite de la réalisation du projet ATMO, nous souhaitions vous soumettre les points essentiels que nous avons rencontré et/ou appris en revenant sur notre expérience.

Importance des spécifications et de la modélisation

Initialement, nous avions uniquement la page donnée en tant que sujet pour guide et au fil des interrogations entre nous et avec le client (joué par les professeurs), nous sommes parvenus à en déduire les fonctionnalités clefs ainsi que pour la forme que le projet devait prendre. Cette première étape fut essentielle puisque c'est sur cette dernière que repose l'intégralité du travail que nous allions réaliser, quelles fonctions seront proposées, comment interagira l'application avec l'utilisateur, quels sont les possibilités d'extension à partir du réalisé… Naturellement, s'en suit l'étape de modélisation du projet, nous nous étions donc répartis les tâches afin que chacun puisse comprendre le fonctionnement d'un module et d'une structure de projet selon l'architecture MVC.
Une fois cette partie terminée, nous procédions toujours à une phase d'échange entre nous pour critiquer/proposer des améliorations sur le réalisé afin d'approcher le plus possible du projet attendu. Etape de validation avec le client oblige, des modifications ont été alors apportées et nous ont permis de comprendre qu'il se pouvait que le client souhaite avoir des informations sur les finalités du projet, par exemple, comment telle fonction sera implémentée puisque grâce à la modélisation, il parvient à avoir une idée plus concrète de ce qui se trame du côté développement du projet.
D'autre part, concernant les patrons de conceptions mis en place, nous avons échangé longuement entre nous pour savoir lesquels étaient les plus intéressants et correspondraient le mieux à notre projet, c'est pourquoi nous nous sommes tournés vers les singletons et via l'utilisation de la « std », des itérateurs.
En effet, une fois cette modélisation terminée, elle constitue une référence d'échange avec le client.

Choix d'un IDE et de tous ses défauts

Une fois la phase de modélisation terminée, venait le temps de proposer nos premiers tests avant même de débuter le développement de l'application afin de s'approcher du modèle de travail de TDD[1] et pour cela, un peu rapidement, nous avons fait le choix d'utiliser Visual Studio puisqu'il permettait de réaliser ses derniers assez simplement et cela nous semblait alors être une aubaine puisque nous pourrions avancer plus vite dans notre projet. Cependant, nous n'avions pas vraiment mesuré l'ampleur de qu'allait avoir ce choix puisqu'en choisissant cet IDE, nous allions devoir nous accommoder de ses défauts et de nombreux problèmes de versions dus aux différences entre les logiciels disponibles au département et notre propre matériel. En effet, nos séances de travail commençaient régulièrement par de la correction de bugs issus des fichiers de configuration de Visual Studio, même si nous veillons à ne pas les versionner.
Ce fut, très honnêtement, un peu usant, mais disons que nous avons fini par nous en accommoder en ne travaillant quasi-exclusivement plus que sur notre matériel personnel, sans oublier de vérifier que les tests passaient aussi sur le matériel du département.
Finalement, nous pensons que si nous nous étions orientés vers une stratégie plus simple de Makefile avec un Framework de test comme dans les projets du premier semestre, peut-être aurions-nous gagner du temps et il est donc important de considérer cette option comme partie intégrante du projet.

Répartition du travail et organisation d'une équipe

Dès le début, l'équipe s'était fixée comme objectif de toujours relire le code produit par les autres membres et pour permettre cela simplement tout en assurant le versionnage du projet, nous avons choisi d'utiliser GitHub. En effet, ce dernier nous permettait simplement de travailler chacun de notre côté, en conservant la possibilité à tout moment d'utiliser les modules développés par nos collègues.

[1] Test Driven Development : Développement piloté par les tests, méthode de développement logiciel vue en cours.

D'autre part et au niveau de la répartition du travail dans le cadre global du projet, nous avons essayé de donner des modules à chaque personne/groupe afin qu'il puisse avoir l'intégralité du champ d'action sur ce qu'il faisait pour s'approcher le plus possible d'un code responsable uniquement de son domaine. En effet, nous souhaitions respecter les principes d'encapsulation c'est pourquoi nous avons évité de mettre en place un grand nombre de méthodes d'accès comme les setters et getters, par exemple. Pour revenir sur la répartition, il fut difficile pour nous de répartir très équitablement le travail, certains en avait toujours plus que d'autre mais pour rééquilibrer la balance, nous avons veillé à toujours aider dès que possible nos camarades qui rencontraient des difficultés.

Retour global sur le projet

Pour donner notre avis sur l'ensemble du projet, la façon dont il est proposé et les opportunités qu'il offre nous ont plus puisque nous avions alors l'occasion de véritablement collaborer et de se rapprocher un peu plus du véritable monde du travail dans lequel chacun des points cités ci-dessus ainsi que la qualité du code sont primordiaux. Si nous devions refaire l'expérience, peut-être que nous accorderions un peu plus de temps au choix de l'environnement de développement et aux choix de modélisation (les classes singletons, les hiérarchies etc..) mais dans l'ensemble, il est assez plaisant de constater que le projet a pu aboutir à partir d'un simple paragraphe de souhaits du client.