Université de Rouen

Master 1 Génie de l'Informatique Logicielle

Date : Octobre 2015 - Mai 2016

Rapport du projet
UML Reverse

Auteurs :
Anthony Godin
Florian Inchingolo
Nabil Belkhous
Nicolas Meniel
Saad Mrabet
Stephen Cauchois
Yohann Henry

Client :
Stéphane Hérauville

Table des matières

1 Introduction

Le projet **UML Reverse** vise à développer un logiciel permettant de construire des diagrammes UML graphiquement. Les diagrammes sont générés dans un projet. Ils sont éditables ergonomiquement avec la souris. Les diagrammes gérés pour cette premiere version sont les diagrammes de cas d'utilisation et de classe. Toutes les classes créées dans un diagramme sont enregistrées dans un projet. Le but est de pouvoir à l'avenir relier les entités d'un diagramme de classe avec un diagramme de séquence. L'application offre plusieurs fonctionnalités :

— créer et éditer des diagrammes dans un projet ;
— sauvegarder et charger des projets ;
— exporter un diagramme en image ou l'imprimer ;
— importer un fichier PlantUML en diagramme dans un projet ;
— exporter un diagramme en fichier PlantUML ;
— importer un fichier Java ou un paquetage Java en diagramme dans un projet.

2 Gestion du projet

2.1 Organisation

Ce projet s'est déroulé en deux phases. La première, la **phase d'analyse**, est la conception des documents nécessaires pour mener le projet à bien. La deuxième phase est la **phase de développement**, qui consiste à concevoir l'application issue des besoins et des documents définis lors de la phase d'analyse. Ces phases se sont déroulées en utilisant des méthodes de développement agiles pour s'assurer de rendre un projet dans les temps en respectant le plus possible les attentes du client.

2.1.1 Phase d'analyse

Les documents ont été rédigés de octobre 2015 à janvier 2016. Nous avons commencé par la STB pour définir les attentes du projet puis nous avons rédigé les autres. Nous nous sommes divisés en équipes de deux personnes pour chaque document. Chaque document était relu par les autres membres de l'équipe afin de les perfectionner.

2.1.2 Phase de développement

Le développement a débuté le 28 janvier. Nous avons défini au début de projet plusieurs rôles, que nous avons affiné au fur et à mesure de la conception du projet :

Anthony Godin	Chef de projet et responsable client
Florian Inchingolo	Responsable de l'architecture du parseur
Yohann Henry	Responsable de l'architecture du modèle
Nabil Belkhous	Responsable des tests
Nicolas Meniel	Développeur
Stephen Cauchois	Responsable de l'architecture graphique
Saad M'Rabet	Développeur

À noter que chaque membre de l'équipe du projet était aussi développeur.

Scrum Nous avons choisi d'utiliser une méthode de développement agile et plus particulièrement Scrum pour répondre le mieux possible aux besoins du client en « l'intégrant » dans notre équipe. Grâce à cette méthode nous pouvions régulièrement le solliciter pour modéliser le produit de la façon la plus fidèle possible à ce qu'il souhaitait.

Nous avions prévu trois itérations finalisées par trois livrables qui composaient le produit demandé. Au début de chaque itération, nous faisions une *préparation de Sprint* afin de planifier les tâches nécessaires ainsi que de les répartir. Nous utilisions un *Sprint Backlog* avec Trello afin d'aider à la préparation du sprint. Nous commencions par la suite le développement qui était vérifié par les tests unitaires.

Une mêlée de 30 minutes était organisée une fois par semaine (le lundi à 10 h) afin que toute l'équipe soit au courant de l'avancement de chacun. Nous detections grâce à elle les difficultés rencontrées et

trouvions des solutions. Nous en profitions pour également distribuer les tâches non attribuées. Nous organisions également des rendez-vous ponctuels de développement (un rendez-vous juste après la mêlée hebdomadaire et une le jeudi à 14 h).

Plusieurs livrables ont été livrés afin de faire valider le travail par le client pendant le développement. Avant chaque livrable, nous lancions une phase de tests pour vérifier les tests fonctionnels du CDR afin de valider toutes les exigences du client. Nabil Belkhous s'occupait de remplir les recettes et était toujours accompagné d'une personne ou deux pour effectuer tous les tests fonctionnels.

Chaque itération était terminée par un *Sprint Review* afin de vérifier l'avancement concret du projet dans son ensemble, puis d'une *Sprint Retrospective* pour identifier les possibles difficultés et d'identifier les points forts et faibles de l'équipe lors de cette itération.

2.2 Interaction avec le client

Nous disposions d'un responsable client qui avait la responsabilité de maintenir le contact avec le client durant toute la phase du projet. Il écrivait un compte-rendu après chaque réunion qu'il envoyait à l'équipe et résumait lors des mêlées.

2.2.1 Phase d'analyse

Une réunion était prévue toutes les semaines. Le but était de comprendre exactement ce que le client voulait et de rédiger avec lui les spécifications des besoins, ce qui nous a permis la rédaction des autres documents ainsi que la prévision du coût et l'organisation du projet.

2.2.2 Phase de développement

Une réunion était prévue au minimum toutes les deux semaines avec M. Hérauville. Elle permettait :
- de poser des questions précises sur ses envies avant de développer une fonctionnalité ;
- de lui faire une démonstration après avoir développé une fonctionnalité pour avoir son avis et éventuellement corriger si besoin.

L'objectif de ces réunions était de se rapprocher le plus possible de l'envie du client sans perdre de temps et d'énergie (agilité). Chaque itération se finalisait par un livrable que le client validait.

2.3 Gestion des risques

2.3.1 Mauvaise estimation de la taille du projet

Lors de la phase d'analyse, nous nous sommes rendu compte que le projet était trop volumineux pour le temps que l'on nous accordait. Il fallait étendre les fonctionnalités de l'application sur cinq diagrammes UML différents. Nous avons dû négocier avec le client pour limiter le nombre de diagrammes et finalement se limiter à deux diagrammes obligatoires. Même après ces négociations, le projet reste tout de même de taille avec environ 250 classes.

2.3.2 Changement de chef de projet

Lors de la phase d'analyse, nous avions élu Florian Inchingolo comme chef de projet, mais en janvier, il a eu des problèmes personnels qui ont restreint ses disponibilités. Son manque de présence au niveau de l'organisation rendait la gestion du projet difficile. C'est pourquoi, avec l'accord de l'équipe, nous avons décidé de réagir vite en changeant de chef de projet et avons élu Anthony Godin à ce poste.

2.3.3 Défaut de l'architecture du modèle

Au début de la phase de développement, suite à des réunions avec M. Hérauville, nous nous sommes rendu compte que notre architecture pour le modèle ne correspondait pas totalement aux besoins exprimés. Nous avions mal tenu compte de sa volonté de poursuivre le projet avec une autre équipe avec l'ajout de certaines fonctionnalités, ce qui aurait pu poser problème aux futures équipes. Nous avons donc décidé de reprendre la conception du modèle au début de la phase de développement. Au même moment, de nouvelles idées de fonctionnalités ont été proposées par le client, comme par exemple l'idée de pouvoir enregistrer les classes dans un projet et les réutiliser dans un diagramme de séquence. Cela

nous a destabilisés. Ces nouvelles fonctionnalités ont entraîné des modifications du modèle de projet et de diagramme de classe. Nous avons donc dû nous réorganiser. Au lieu de faire le modèle et ensuite la vue, nous avons commencé le développement de la vue en simulant un modèle, le temps de refaire son l'architecture.

2.3.4 Adaptation au travail d'équipe et au projet

Nous n'avions jamais travaillé dans un groupe aussi important et nous ne nous connaissions pas particulièrement, ce qui nous a créé quelques problèmes lors des premières semaines de développement avec de mauvaises estimations de délais. Toutes les semaines, lors des mêlées, nous constations souvent des retards. Nous nous sommes vite rendu compte qu'on ne pourrait pas continuer comme ça. Nous sommes devenus moins laxistes, plus attentifs au respect des engagements. Mais ça n'a pas suffi pour rattraper le retard pris pour le premier livrable. Nous avons donc décidé de mettre en pause une tâche extérieure au livrable pour avoir un développeur supplémentaire pour rattraper le retard. Cette solution a fonctionné et nous avons pu rendre le livrable à temps et le valider par le client. Une marge était prévue lors de la deuxième itération, ce qui nous a permis de finir les autres tâches en retard.

2.4 Tests

Les tests logiciels ont été un élément de poids lors de la réalisation de notre projet pour mettre en évidence ses défauts. Nous avons conçu des **tests automatiques** (tests unitaires) et rédigé sous forme de recettes nos **tests fonctionnels** (tests manuels). Ils ont joué un rôle majeur pour assurer un niveau de qualité défini en accord avec le client.
La planification de la majorité des tests fonctionnels a été définie lors de la première phase (conception des documents).

Les tests unitaires ont été réalisés à l'aide du framework JUnit. Avant chaque livraison, et après les tests unitaires de chaque partie du livrable, des **tests de validation** ont été faits. Ils étaient en deux parties :

- les **tests de validation fonctionnels** qui vérifiaient que les différents modules réalisés correspondaient aux exigences du client ;
- les **tests de validation solution** ou de validation par cas d'utilisation, qui consistaient à exécuter des scénarios de tests regroupant plusieurs modules (exigences du client).

Au moment de la livraison, les **tests d'acceptation** ont été faits avec le client (démonstration) pour s'assurer que le livrable convenait bien à ses exigences. Les défauts détectés par le client ont été pris en compte et corrigés.

Chaque membre de l'équipe a participé à ces tests, que ce soit la rédaction des recettes, les tests unitaires ou le remplissage des recettes. Pour corriger et partager les défauts découverts, lors des tests, nous avions mis en place sur Google Drive un tableur où nous les recensions, avec le scénario à exécuter pour les reproduire.

3 Choix technologiques

- Git : Il été notre premier choix. Certains d'entre nous connaissaient déjà ce logiciel de gestion de versions décentralisé. Il est essentiel d'avoir ce type de logiciel pour travailler en équipe. C'est lui qui s'occupe de fusionner et d'organiser le travail de tous dans un même projet.
- Slack : Plate-forme de communication collaborative ainsi qu'un logiciel de gestion de projets. C'est grâce à lui que nous communiquions dans divers chats qui ne concernaient que ce projet. De plus, il offre des nombreux services utiles comme la connectivité avec Trello et Git, ce qui nous permettait de recevoir des notification sur Slack dès qu'une version était poussée sur Git ou une tâche éditée sur Trello.
- Trello : Outil de gestion de projet en ligne. Il nous a permis d'énumérer nos taches en y fixant des dates, des personnes et des états (en cours, fait, en test, etc.)
- Apache Maven : Outil pour la gestion et l'automatisation de production des projets logiciels Java. Maven nous a permis de créer notre architecture globale et d'utiliser certains outils comme JUnit ou Antlr sans devoir les télécharger nous-mêmes manuellement sur nos machines.

— Java 8 et JavaFX : Java était imposé dans le projet. Nous avons utilisé Java 8, la dernière version de Java, car celle-ci offre de nouvelles fonctionnalités intéressantes. Le choix de JavaFX a été pris car nous ne connaissions pas beaucoup d'autres choix graphiques, et celui-ci était présenté comme le successeur de Swing que nous avions étudié en licence.
— Antlr : Framework de construction de compilateurs. Il nous a servi lors du parsage Java et PlantUML.

4 Introspection

Le projet est maintenant terminé. Analysons ce que nous changerions si nous devions recommencer le projet. Quels sont les choix que nous aurions changé ?
Nous avons à présent l'expérience du projet, de l'équipe et surtout des connaissances du Master 1 GIL qui nous ont fait défaut durant l'analyse du projet.

Au niveau de l'**architecture**, nous avons constaté que certains choix ont révélé des faiblesses à terme. Nous avons donc noté dans un document destiné à la prochaine équipe plusieurs points pour simplifier la reprise du projet, mais aussi pour reprendre l'architecture et, s'ils le souhaitent, l'optimiser afin de réduire sa complexité.
Dans ce document, nous proposons l'usage de Dot qui avait été prévu au début du projet mais que le temps nous a contraint à ne pas utiliser.
De plus, nous expliquons que la sauvegarde du projet serait bien plus performante au format XML plutôt qu'au format binaire qu'elle utilise actuellement. Nous avions fait ce choix car nous ne connaissions pas encore très bien la sérialisation d'objets en format XML.

Concernant les **technologies utilisées**, nous sommes satisfaits de nos choix. Le seul choix que nous pourrions regretter serait JavaFX. Au départ, nous avions constaté de nombreux avantages à cette bibliothèque. Le glisser-déposer, le zoom ainsi que le CSS nous ont permis de gagner beaucoup de temps. Néanmoins, aujourd'hui, nous pensons avoir investi dans une bibliothèque assez peu utilisée en entreprise et nous aurions préféré choisir une technologie plus utile à notre cursus.

L'**organisation de l'équipe** serait légèrement différente durant la première itération. Au début du projet, nous ne nous connaissions pas très bien entre nous. Nous avions fait connaissance en exposant nos qualités et nos défauts. Mais c'était notre premier gros projet et nous ne connaissions pas forcément nos potentiels ni nos lacunes en terme de travail d'équipe. Nous aurions donc pris des choix d'organisation d'équipe plus réfléchis et pertinents maintenant que l'on se connaît mieux.
Nous aurions dû aussi éviter de laisser seuls des membres de l'équipe sur de grosses tâches comme ça a été le cas pour le modèle de diagramme de classe. Le fait de travailler au moins à deux permet d'avoir quelqu'un sur qui se reposer, d'avoir une personne pour prendre du recul et d'avoir certaines fois deux angles de réflexion différents sur les problèmes complexes.

L'**organisation du projet** serait aussi plus efficace maintenant. En raison de la reconception du modèle au début du développement (expliquée précédement dans la gestion des risques), nous avions séparé l'équipe pour éviter un blocage. La moitié de l'équipe a travaillé à la conception de la vue sans modèle pour plus tard l'intégrer.
Si nous devions rencontrer de nouveau ce problème, nous occuperions une plus grosse partie de l'équipe à la conception du nouveau modèle afin d'éviter des intégrations aussi compliquées.
De plus, notre expérience du projet nous donnerait aujourd'hui une meilleure analyse des temps de conception de chaque partie, ce qui permettrait d'obtenir un diagramme de Gantt plus robuste.

5 Conclusion

5.1 Bilan du projet

Nous avons dû restreindre le sujet car au départ, il était trop volumineux. Il fallait pouvoir éditer 5 types de diagrammes UML différents. Nous nous sommes restreints au deux plus importants après

négociation avec le client. Mais **les objectifs prévus ont été réalisés**. Une bonne partie des classes codées sont réutilisables, comme le voulait le client. Les futures fonctionnalités à ajouter seront donc simplifiées, que ce soit au niveau de la vue ou du modèle.

5.2 Bilan du personnel

Nous avons découvert la gestion de projet à travers une équipe composée plus de deux ou trois personnes, ce qui ne nous était jamais arrivé. De plus, nous n'avions jamais travaillé sur un projet aussi conséquent, ce qui nous a amenés à mettre en pratique pour la première fois une méthodologie Agile avec Scrum. Ce projet nous a permis d'améliorer nos compétences en organisation (travail d'équipe et architecture logicielle), ainsi que sur le plan technique (JavaFX, Java 8, Maven). Nous avons aussi appris de nous-mêmes, nous avons une meilleure estimation de temps des tâches et une plus grande expérience en travail d'équipe.