Plan de développement
Projet UML Reverse

26 janvier 2016

Version :	1.0
Date :	26 janvier 2016
Rédigé par :	Florian Inchingolo Anthony Godin
Relu par :	Florian Inchingolo Anthony Godin

Mises à jour

Version	Date	Modifications réalisées
1.0	21/01/2016	Corrections avant la présentation
0.1	10/01/2016	Début de la rédaction du PDD

Table des matières

1 Contexte du projet

1.1 Fiche du projet

Dans le cadre de la formation de première année en Master en Génie Informatique Logiciel (GIL) de l'UFR de Sciences et Techniques de Rouen, un projet annuel en équipe de 6 ou 7 personnes est proposé parmi une liste imposée.

Vision

Le but du projet est d'obtenir un éditeur de diagrammes UML2. Ce diagramme pourra être obtenu à partir d'un fichier PlantUML, à partir d'un ensemble de sources Java ou être créé de toutes pièces via l'éditeur. Chaque entité du diagramme doit pouvoir être déplacée ou stylisée.

Objectifs

L'application devra :
— être modulaire et documentée ;
— être ergonomique ;
— respecter UML2.

1.1.1 Matrice des leviers

	Souple	**Moyen**	**Rigide**
Coût / Temps	X		
Durée			X
Qualité			X
Portée		X	

1.2 Phases de développement

Ce projet se déroule en deux phases imposées. La première, la **phase d'analyse**, est la conception des documents nécessaires pour mener le projet à bien. Cette phase se rapproche des méthodes de développement classiques. La deuxième phase est la **phase de développement**, qui consiste à développer l'application issue des besoins et des documents définis lors de la phase d'analyse. Cette phase doit se dérouler 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 Terminologie

Se référer au document terminologie.

3 Préparation du projet

3.1 Organisation de l'équipe

Nous avons défini au début de projet plusieurs rôles, que nous avons affiné au fur et à mesure de la conception du projet :

Florian Inchingolo	Chef de projet
Anthony Godin	Responsable client
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

A noter, chaque membre de l'équipe du projet est aussi bien entendu développeur.

3.2 Méthodologie de développement

Pour mener à bien la phase de développement du projet, nous allons utiliser la méthode Scrum. Cette phase sera donc séparée en plusieurs itérations contenant chacune un livrable du projet.

Au début de chaque itération, nous ferons un *Sprint Planning* afin de planifier les tâches nécessaires ainsi que de les répartir. Nous utiliserons un *Sprint Backlog* afin d'aider à la préparation du sprint lors des deux dernières itérations. Ensuite, pendant une durée de 2 à 3 semaines, nous serons en phase de développement. Celle-ci sera dirigée par les tests, mais les tests seront écrits après le développement de chaque classe. Les tests unitaires seront effectués par le développeur. Au début de chaque rendez-vous de développement (entre une à trois fois par semaine, dépendant des disponiblités de l'équipe au complet), nous ferons une mélée afin que toute l'équipe soit au courant de l'avancement de chacun, et ainsi déterminer une possible surcharge de travail de chacun, ou de préciser les tâches prévues par chacun lors de ce rendez-vous. Ces rendez-vous ne seront pas les seuls moments où les membres de l'équipe développent, mais c'est un moment où toute l'équipe est présente. Lors de la *phase de test* de chaque itération durant une semaine, chaque développeur teste les fonctionnalités et l'intégration d'une partie qu'il n'a pas contribué, afin de déterminer les possibles erreurs plus simplement (un regard "externe" sera mieux à même de trouver les problèmes fonctionnels). Cette phase sera terminée par un *Sprint Review* afin de vérifier l'avancement concret du projet dans son ensemble, puis d'un *Sprint Rétrospective* pour identifier les possibles difficultés et d'identifier les points forts et faibles de l'équipe lors de cette itération.

3.3 Outils utilisés

Nous utilisons Git afin de pouvoir gérer les contributions de chacun correctement. Chaque tâche et sous-tâche est affiché au sein de Trello, un bloc-notes collaboratif. Tous les membres de l'équipe peuvent ainsi voir l'avancement de chacun, s'attribuer des tâches, ou aider les autres quand ceux-ci ont trop de tâches qui leur sont attribuées. Pour la communication au sein de l'équipe, nous utilisons Slack.

3.4 Recherche de minimisation des risques

Lors de l'analyse des risques, nous avons mis à jour plusieurs risques qui pouvaient être problématiques. Pour éviter que ceux-ci puissent nuire au projet, nous avons fait des recherches afin de maîtriser les questions n'ayant pas encore de réponse.

Par exemple, pour l'analyse lexicale du langage Java, nous utilisons Antlr, mais personne ne maîtrisait le lexeur. Nous avons donc codé une preuve de concept en amont de la phase de développement montrant qu'il est facilement possible d'analyser un code source Java pour en afficher l'arbre syntaxique.

Un deuxième risque était l'utilisation de JavaFX. Même si l'un d'entre nous avait déjà utilisé cette technologie, nous n'étions pas tous familiers avec celle-ci. Nous avons donc décidé de créer un programme d'exemple, afin de voir les possibilités de l'API.

Pour ce qui est de la gestion du glisser-déposer, nous avons trouvé et analysé un programme d'exemple afin de ne pas avoir de mauvaise surprise lors du développement.

Enfin, pour les relations, nous avons défini un système à base de points afin d'amoindrir grandement la complexité de programmation de ces dernières.

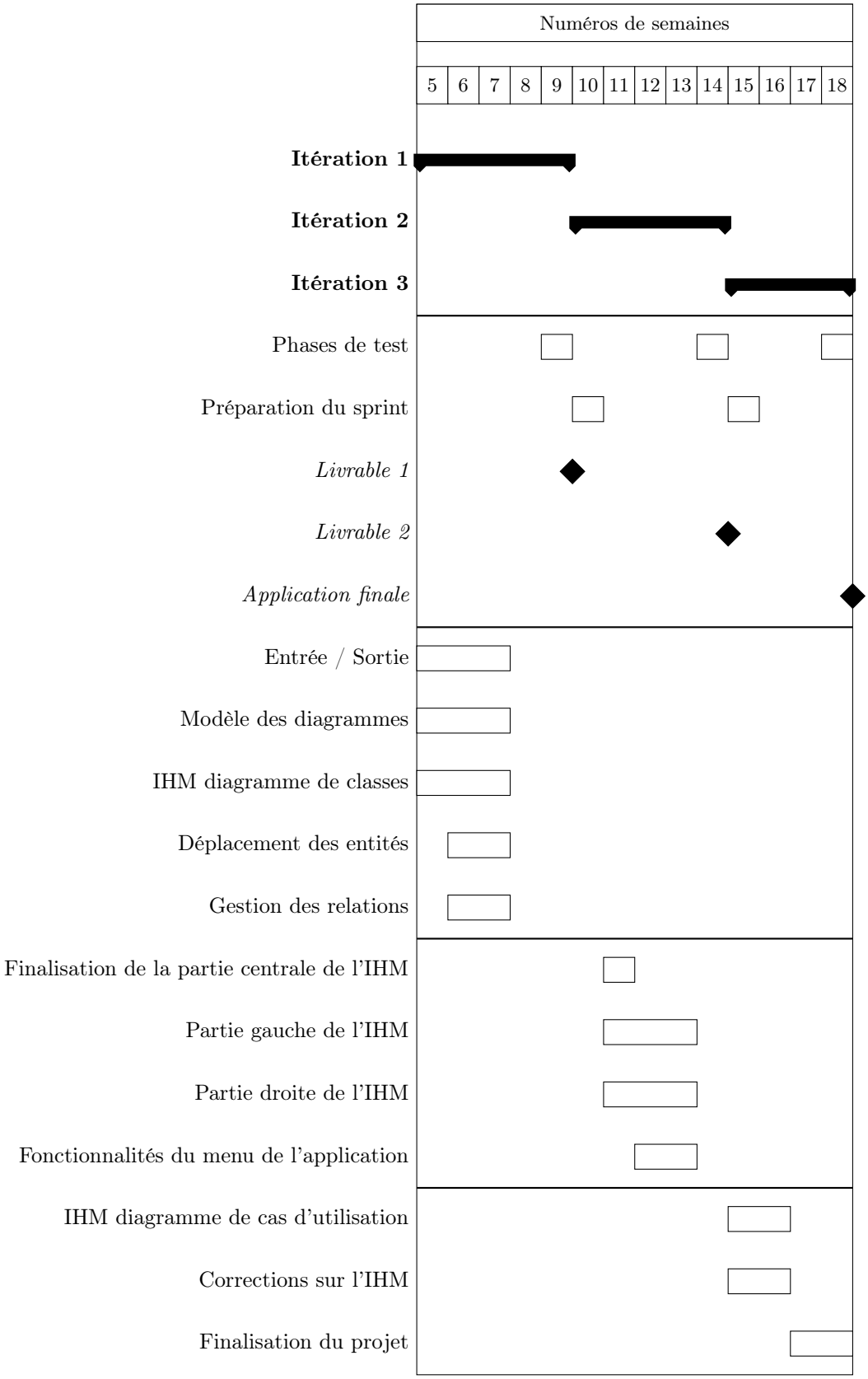
4 Planification du projet

Pour ce projet, nous avons estimé qu'il était bénéfique de proposer 3 livrables. En effet, nous comptons présenter une première version contenant seulement ce qui est nécessaire à un début d'édition du diagramme de classes, afin de s'atteler à tous les risques possibles de blocage dès le début. Ensuite, nous proposerons au client notre vision de l'interface graphique pour le deuxième livrable. Nous avons prévu peu de tâches lors de la troisième itération (diagramme de cas d'utilisation) afin de pouvoir apporter les corrections demandées par le client sur l'IHM s'il y en a, de s'assurer que l'on peut s'engager sur ce que l'on propose, ou dans le cas où le projet se déroule mieux que prévu, de pouvoir préparer de nouvelles fonctionnalités.

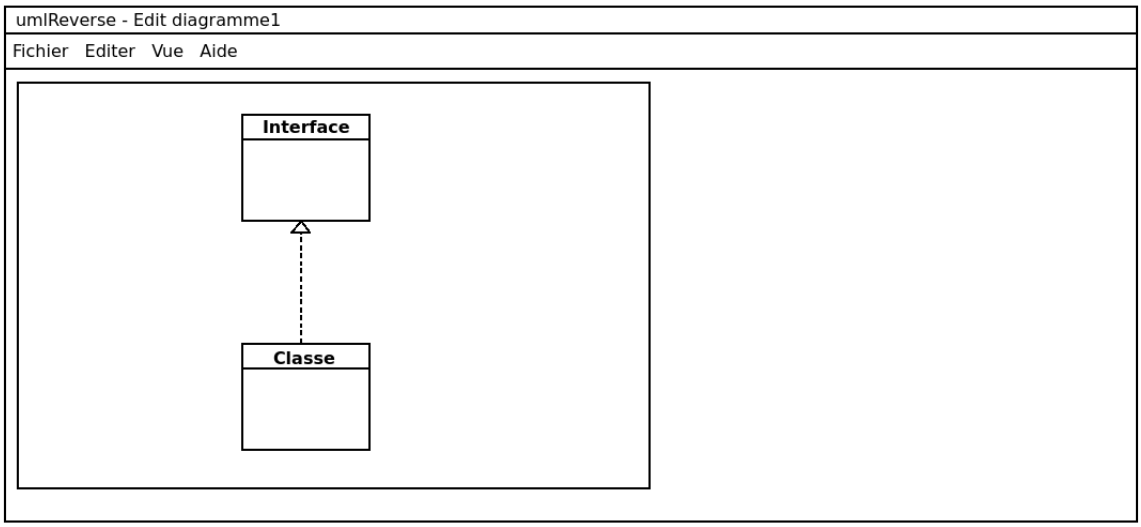
4.1 Présentation et dates de disponibilités des livrables

— **Livrable 1** : Disponible à la fin de la semaine 9, ce livrable sera la base de notre projet. Celui-ci comprendra une partie des fonctionnalités essentielles à un éditeur, ainsi qu'un modèle correspondant à la version finale du projet. Le temps de développement nécessaire à ce livrable a été estimé entre 115 à 145 heures.
— **Livrable 2** : Disponible à la fin de la semaine 14, le but de ce livrable sera d'implémenter l'interface homme-machine la plus fidèle à l'interface attendue à la fin du projet. Le temps de développement nécessaire à ce livrable a été estimé à environ 130 heures.
— **Livrable 3** : Disponible à la date limite du rendu de projet (semaine 18), l'objet de ce livrable sera l'ajout du diagramme de cas d'utilisation, et si possible, des fonctionnalités supplémentaires. Le temps de développement nécessaire à ce livrable a été estimé à environ 110 heures.

4.2 Diagramme de Gantt



4.3 Itération 1



Maquette de l'interface homme-machine de l'itération 1.

Le but de cette itération est de produire un premier livrable comprenant les fonctionnalités minimales permettant d'utiliser l'application. Ce livrable permettra seulement de charger un diagramme à partir d'un fichier PlantUML avec un style ou non ou d'un code Java, puis de pouvoir sauvegarder le résultat, récupérer le PlantUML résultant ou son style. Ce dernier comprendrait seulement la position des entités. Comme cela, nous effectuons les tâches les plus risquées dès le début, afin de minimiser l'impact d'autres évènements attendus ou non (comme les phases de projets ou la possibilité de la perte d'un membre de l'équipe) sur les prochaines itérations, ainsi que d'éviter un "blocage" lors des itérations suivantes.

4.3.1 Fonctionnalités disponibles sur ce livrable

- **IHM_60** : Exporter un fichier de paramètres
- **IHM_100** : Importer/charger un diagramme UML écrit en PlantUML compatible dans un projet
- **DIA_60** : Générer un diagramme de classes par reverse engineering sur du code Java
- **MOD_40** : Déplacer une entité

4.3.2 Définition des tâches

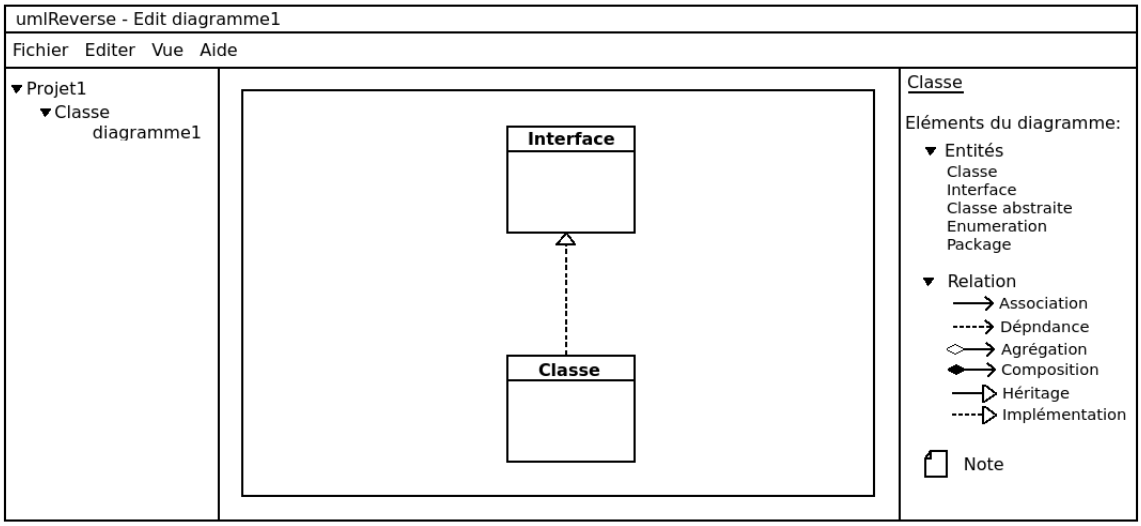
- **Entrée / Sortie** : Création des parseurs permettant de créer un diagramme de classes à partir d'un code Java, un diagramme compatible à partir d'un fichier PlantUML, ainsi que de charger un fichier de style à appliquer sur le diagramme. En parallèle, création des visiteurs responsables de la sauvegarde et des exports.
- **Modèle des diagrammes** : Écriture du modèle complet des diagrammes, comprenant le code commun à tous les diagrammes ainsi que les classes spécifiques au diagramme de classes, de cas d'utilisation et de séquence.
- **IHM diagramme de classe** : Développement du composant permettant l'édition du diagramme de classes, avec une fenêtre ayant des options minimales comme le chargement et la sauvegarde. Sur cette itération, la seule action possible lors de l'édition sera de pouvoir déplacer les entités.
- **Déplacement des entités** : Gestion du glisser-déposer sur une entité quelconque dans le composant du diagramme.
- **Gestion des relations** : Développement de l'affichage et le déplacement de toutes les relations disponibles sur les diagrammes prévus. Celles-ci ne seront pas éditables sur ce livrable.

4.3.3 Attribution des tâches

- **Entrée / Sortie** : Florian Inchingolo

— **Modèle des diagrammes** : Yohann Henry (responsable), Nicolas Meniel, Nabil Belkhous et Saad M'Rabet
— **IHM diagramme de classe** : Stephen Cauchois (responsable) et Anthony Godin,
— **Déplacement des entités** : Stephen Cauchois
— **Gestion des relations** : Anthony Godin

4.4 Itération 2



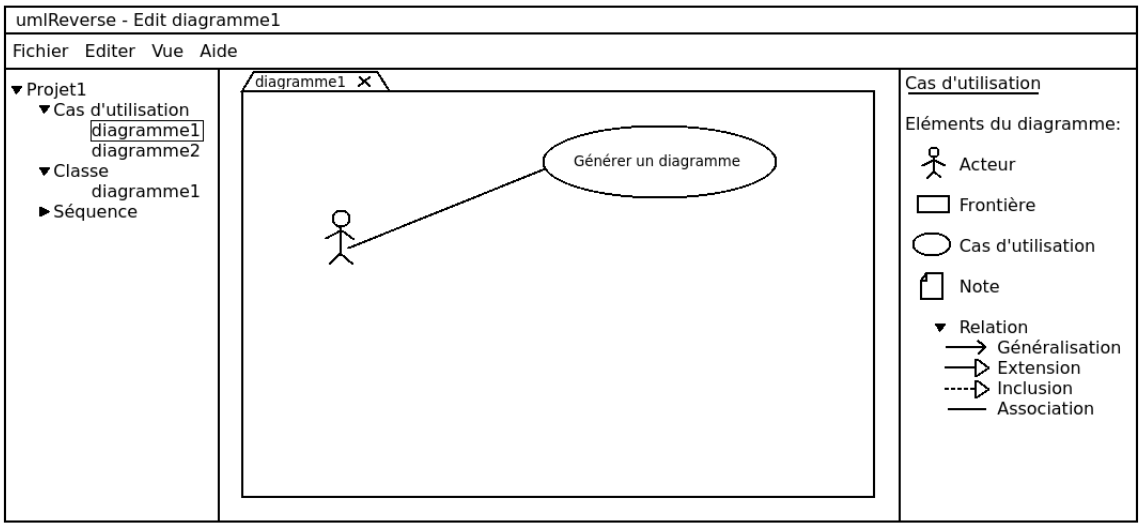
Maquette de l'interface homme-machine de l'itération 2.

4.4.1 Fonctionnalités disponibles sur ce livrable

— **DIA_20** : Créer un nouveau diagramme de classes
— **IHM_*** : Toutes les fonctionnalités indispensables liées à l'IHM
— **IHM_80** : Exporter un diagramme au format image ou PDF
— **MOD_*** : Toutes les fonctionnalités indispensables liées à la modification d'un diagramme
— **CLA_*** : Toutes les fonctionnalités indispensables liées au diagramme de classes

Une définition des tâches plus précise ainsi que son attribution sera l'objet de la préparation du sprint de l'itération 2, semaine 10.

4.5 Itération 3



Maquette de l'interface homme-machine du livrable final.

4.5.1 Fonctionnalités disponibles sur ce livrable

— **DIA_10** : Créer un nouveau diagramme de cas d'utilisation
— **USE_*** : Toutes les fonctionnalités liées au diagramme de cas d'utilisation

Une définition des tâches plus précise ainsi que son attribution sera l'objet de la préparation du sprint de l'itération 3, semaine 15.

4.6 Evaluation de la charge de travail

Les estimations ci-dessous sont exprimées en heures, puis en semaine/homme (s/h).

— **Entrée / Sortie** : Environ 20 heures, soit 3s/h.
— **Modèle des diagrammes** : Entre 40 et 60 heures, soit de 6 à 9 s/h.
— **IHM diagramme de classes** : Entre 30 et 50 heures, soit de 4 à 7s/h.
— **Déplacement des entités** : Environ 10 heures, soit 1.5s/h.
— **Gestion des relations** : Environ 15 heures, soit 2s/h.
— **Finalisation de la partie centrale de l'IHM** : Environ 15 heures, soit 2s/h.
— **Partie gauche de l'IHM** : Environ 40 heures, soit 6s/h.
— **Partie droite de l'IHM** : Environ 40 heures, soit 6s/h.
— **Fonctionnalités du menu de l'application** : Environ 35 heures, soit 5s/h.
— **Corrections sur l'IHM** : Environ 20 à 30 heures, soit 3s/h. Dépend des corrections demandées.
— **IHM diagramme de cas d'utilisation** : Environ 20 à 30 heures, soit 3s/h.
— **Finalisation du projet** : Dépend fortement de l'avancement du projet au début de la troisième itération. Peut être estimé à environ 50 heures, soit 8s/h.

4.7 Disponibilités de l'équipe

Après réunion d'équipe, chaque membre se rend disponible pour le projet environ 6 ou 7 heures par semaine. Ce temps sera réduit à environ 5 heures lors de la troisième itération à cause des besoins de temps dans les autres matières qui vont augmenter. Ce qui nous fait environ 200 heures de disponble lors de chacune des deux premières itérations, et 150 heures lors de la troisième. Un tiers de ce temps sera consacré aux activités annexes au développement de l'application, comme les réunions client, les phases de tests fonctionnels, les réunions Scrum ou la préparation du sprint.