Cahier des recettes
Projet UML Reverse

3 mai 2016

Version :	0.1
Date :	3 mai 2016
Rédigé par :	Nabil Belkhous Anthony Godin
Relu par :	Anthony Godin Florian Inchingolo Nicolas Meniel

Table des matières

1 Introduction

Ce document a le but de définir les tests nous permettant de prouver que le projet en cours d'élaboration respecte bien le contrat en accord avec le client.
Chaque cas d'utilisation a un test défini, ainsi que chaque exigence fonctionnelle.

2 Cas d'utilisation et exigences fonctionnelles

Voir le document STB.

3 Terminologie et sigles utilisés

Se référer au document Terminologie.

Test de modification d'élément/entité :
Ici on veut dire tester toutes les modifications possibles du contenu de l'élément/entité défini dans la STB. Ajout, suppression, cachement, affichage du contenu.

(Action) de toutes les manières possibles :
Nous n'avons pas encore défini toutes les manières de réaliser une action graphiquement. Actuellement des boutons sont prévus, mais le client pourra être plus précis au moment où nous aurons une version à lui présenter. Il y aura un manuel d'utilisation qui listera toutes les manières possibles d'exécuter une action. Il faudra se référer à ce document pour tester toutes les manières possibles d'exécuter une action.

4 Environnement de test

Le projet est produit avec Maven. Maven nous servira de gestionnaire de test. Chaque fichier de classe a un fichier de test qui lui est associé. Le code source sera dans le dossier main et les tests dans un dossier test. La compilation d'une classe relancera tous les tests qui lui sont associés. Bien sûr, une fois qu'une classe est compilée, elle le sera à nouveau seulement si elle est modifiée, ce qui forcera à refaire tous les tests. JUnit est intégré dans Maven. Voici l'arboresence des fichiers.

```
src
├── main
│   └── java
│       └── com
│           └── monprojet
│               └── Class.java
└── test
    └── java
        └── com
            └── monprojet
                └── ClassTest.java
```

5 Responsabilités et stratégie

— **Responsable des tests** : Nabil Belkhous

Le responsable des tests est la personne chargée de vérifier que tous les tests annoncés dans ce document sont validés.

Les tests seront écrits par l'ensemble de l'équipe. Chaque membre écrit les tests du code qu'il produit. Les développeurs ne coderont qu'une classe à la fois. Toute classe devra être testée et approuvée avant d'être définie comme terminée. Le développeur pourra ensuite (et seulement ensuite) passer au développement de la classe suivante.
Les tests seront effectués à l'aide de JUnit et Maven de la façon décrite dans la section « Environnement de test ».

6 Tests fonctionnels

On ne notera pas les contraintes de bon sens. Il est évident qu'on a besoin d'un minimum de mémoire quand on enregistre un fichier. On notera aussi qu'on ne peut pas nommer deux fichiers avec le même nom s'ils sont dans le même dossier.

Créer un projet

Identifiant du test :	TEST_1	
Identifiant de l'exigence :	IHM_10	
Objet :	Créer un projet	
Contraintes :		
Action :	Créer un projet de toutes les manières possibles.	
Résultat attendu :	L'application demande à l'utilisateur de donner le chemin où il veut enregistrer son projet sur son ordinateur. Une fois donné, un dossier au chemin donné par l'utilisateur sera créé.	
État :	**Fait**	Le : 03/05/2016

Ouvrir un projet

Identifiant :	TEST_2	
Identifiant de l'exigence :	IHM_20	
Objet :	Ouvrir un projet	
Contraintes :	Il faut que le projet existe.	
Action :	Ouvrir un projet de toutes les manières possibles.	
Résultat attendu :	L'application demande à l'utilisateur de lui donner le chemin du projet. Les dossiers dans les projets sont bien lus et affichés par l'application, ainsi que tous les diagrammes présents dans les fichiers src avec leurs fichiers de paramètres.	
État :	**Fait**	Le : 03/05/2016

Supprimer un projet

Identifiant :	TEST_3	
Identifiant de l'exigence :	IHM_20	
Objet :	Supprimer un projet	
Contraintes :	Le projet doit exister	
Action :	Supprimer un projet de toutes les manières possibles.	
Résultat attendu :	L'application demande à l'utilisateur de sélectionner le projet. Le dossier du projet selectionné est supprimé avec tous ses dossiers fils	
État :	**Fait**	Le : 03/05/2016

Sauvegarder un projet

Identifiant :	TEST_4	
Identifiant de l'exigence :	IHM_20	
Objet :	Sauvegarder un projet	
Contraintes :	Il faut que le projet existe.	
Action :	Sauvegarder un projet de toutes les manières possibles	
Résultat attendu :	L'application demande à l'utilisateur de lui spécifier quel projet il veut sauvegarder si ce n'est pas déjà fait. Les diagrammes (src et paramètres) du projet ont tous été sauvegardés.	
Test supplémentaire :	Si l'utilisateur charge un projet juste après l'avoir sauvegardé, il devrait pouvoir visualiser les mêmes diagrammes qu'au moment de la sauvegarde.	
État :	**Fait**	Le : 03/05/2016

Créer un diagramme

Identifiant :	TEST_5	
Identifiant de l'exigence :	IHM_30 - DIA_10 - DIA_30 - DIA_40 - DIA_50	
Objet :	Créer un diagramme	
Contraintes :		
Action :	Créer un diagramme de toutes les manières possibles	
Résultat attendu :	L'application demande à l'utilisateur de sélectionner dans quel projet il veut enregistrer le diagramme ou en créer un. Elle lui demande également le type du diagramme (cas d'utilisation, séquence ou classe) ainsi qu'un nom. Le nom ne pourra pas être le même qu'un autre diagramme du même type du même projet. Si c'est le cas, l'application le signalera et proposera à l'utilisateur d'écraser le fichier correspondant. Initialise les objets internes à l'application du type de diagramme en question. Le diagramme est affiché (vide) sur l'application (partie graphique). Deux fichiers du nom du diagramme sont créés dans les dossiers src et parameters dans le dossier du type du diagramme du projet séléctionné.	
État :	**Fait**	Le : 03/05/2016

Sauver un diagramme

Identifiant :	TEST_6	
Identifiant de l'exigence :	IHM_30	
Objet :	Sauver un diagramme	
Contraintes :	Il faut que le diagramme existe	
Action :	Sauver un diagramme de toutes les manières possibles	
Résultat attendu :	Les modifications apportés sur le digramme depuis sa dernière sauvegarde ont été enregistrés dans les fichiers lui correspondant	
Test supplémentaire :	Si l'utilisateur charge un diagramme juste après l'avoir sauvegardé, il devrait pouvoir visualiser le même diagramme qu'au moment de la sauvegarde.	
État :	**Fait**	Le : 03/05/2016

Supprimer un diagramme

Identifiant :	TEST_7	
Identifiant de l'exigence :	IHM_30	
Objet :	Supprimer un diagramme	
Contraintes :	Il faut que le diagramme existe	
Action :	Supprimer un diagramme de toutes les manières possibles	
Résultat attendu :	Les fichiers correspondants seront supprimés (dans src et parameters).	
État :	**Fait**	Le : 03/05/2016

Ouvrir un diagramme

Identifiant :	TEST_8	
Identifiant de l'exigence :	IHM_30	
Objet :	Ouvrir un diagramme	
Contraintes :	Le diagramme doit exister	
Action :	Ouvrir un diagramme de toutes les manières possibles	
Résultat attendu :	L'application demande à l'utilisateur de sélectionner un diagramme. Le diagramme est affiché sur l'application (partie graphique)	
État :	**Fait**	Le : 03/05/2016

Charger un fichier PlantUML

Identifiant :	TEST_8	
Identifiant de l'exigence :	IHM_30	
Objet :	Charger un fichier PlantUML	
Action :	Charger un ficier PlantUML de toutes les façons possibles	
Résultat attendu :	Un diagramme est chargé en mémoire et affiché graphiquement tel qu'il est spécifié dans le fichier PlantUML Un fichier de style lui a été affecté.	
État :	**Fait**	Le : 03/05/2016

Zoomer/Dézoomer un diagramme

Identifiant :	TEST_9	
Identifiant de l'exigence :	IHM_40	
Objet :	Zoomer un diagramme	
Contraintes :	Le diagramme doit être ouvert	
Action :	Zoomer/Dézoomer un diagramme de toutes les manières possibles	
Résultat attendu :	On voit visuellement que le zoom/dézoom est appliqué.	
État :	**Fait**	Le : 03/05/2016

Exporter un diagramme en PlantUML

Identifiant :	TEST_12	
Identifiant de l'exigence :	IHM_70 - EXF_50	
Objet :	Exporter un diagramme en PlantUML	
Contraintes :	Il faut que le diagramme existe et soit ouvert	
Action :	Exporter un diagramme en PlantUML de toutes les manières possibles	
Résultat attendu :	L'application demande à l'utilisateur où il veut enregistrer le fichier. Un fichier PlantUML représentant le diagramme est créé où l'utilisateur l'a demandé.	
État :	**Fait**	Le : 03/05/2016

Exporter un diagramme en image

Identifiant :	TEST_13	
Identifiant de l'exigence :	IHM_80	
Objet :	Exporter un diagramme en image	
Contraintes :	Il faut que le diagramme existe et soit ouvert	
Action :	Exporter un diagramme en image de toute les manières possibles	
Résultat attendu :	L'application demande à l'utilisateur où il veut enregistrer l'image. Puis un fichier image est créé où l'utilisateur l'a demandé et représentant visuellement la même chose que le diagramme affiché par l'application	
État :	**Fait**	Le : 03/05/2016

Imprimer un diagramme

Identifiant :	TEST_14	
Identifiant de l'exigence :	IHM_90	
Objet :	Imprimer un diagramme	
Contraintes :	Il faut que le diagramme existe et soit ouvert	
Action :	Imprimer un diagramme de toutes les manières possibles	
Résultat attendu :	L'impression s'exécute correctement	
État :	**Fait**	Le : 03/05/2016

Exporter un diagramme UML en PlantUML

Identifiant :	TEST_15	
Objet :	Exporter un diagramme UML en PlantUML	
Contraintes :	Il faut que le diagramme existe et soit ouvert	
Action :	Exporter un diagramme UML en PlantUML de toutes les manières possibles	
Résultat attendu :	L'application demande à l'utilisateur où il veut enregistrer le dossier. Un dossier représentant le diagramme est créé où l'utilisateur l'a demandé.	
Test supplémentaire :	Importer le dossier exporté visuellement le même diagramme. Au moment de l'export d'un diagramme, sauvegarder le diagramme (celui qui est modélisé). Recharger le diagramme et comparer la première modélisation à la deuxième.	
État :	**Fait**	Le : 03/05/2016

Importer un PlantUML en diagramme UML

Identifiant :	TEST_16	
Identifiant de l'exigence :	IHM_100	
Objet :	Importer un PlantUML en diagramme UML	
Action :	Importer un PlantUML en diagramme UML de toutes les manières possibles	
Résultat attendu :	L'application charge et affiche le diagramme en question	
Test supplémentaire :	Importer le dossier exporté visuellement le même diagramme. Au moment de l'export d'un diagramme, sauvegarder le diagramme (celui qui est modélisé). Recharger le diagramme et comparer la première modélisation à la deuxième.	
État :	**Fait**	Le : 03/05/2016

Reverse

Identifiant :	TEST_17	
Identifiant de l'exigence :	DIA_60 - DIA_70 - DIA_80 - EXF_30	
Objet :	Reverse	
Action :	Générer un diagramme en Reverse de toutes les manières possibles sur un fichier écrit en Java 1.7	
Résultat attendu :	Toutes les classes du code source sont bien représentées dans le diagramme	
État :	**Fait**	Le : 03/05/2016

Modification d'un diagramme

Ici on testera les fonctionnalités de la modification de diagramme visuellement. Les cases correspondant aux fonctionnalités testées et approuvées doivent être remplies par **Fait**.

Tests d'ajout, de suppression, d'affichage, de cachement et de modification d'éléments :

Identifiant :	TEST_18			
Identifiant de l'exigence :	MOD_10 - MOD_20 - MOD_30 - MOD_60 - MOD_80 - MOD_90 - MOD_100			
Diagramme de cas d'utilisation				
Éléments	Ajouter	Supprimer	Afficher/cacher	
Relation de généralisation	**Fait**	**Fait**	**Fait**	
Relation d'extension	**Fait**	**Fait**	**Fait**	
Relation d'inclusion	**Fait**	**Fait**	**Fait**	
Relation d'association	**Fait**	**Fait**	**Fait**	
Note	**Fait**	**Fait**	**Fait**	
Frontière	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
Acteur	**Fait**	**Fait**	**Fait**	
Cas d'utilisation	**Fait**	**Fait**	**Fait**	
Diagramme de classe				
Éléments	Ajouter	Supprimer	Afficher/cacher	
Relation d'implémentation	**Fait**	**Fait**	**Fait**	
Relation d'héritage	**Fait**	**Fait**	**Fait**	
Relation de composition	**Fait**	**Fait**	**Fait**	
Relation d'agrégation	**Fait**	**Fait**	**Fait**	
Relation d'association	**Fait**	**Fait**	**Fait**	
Relation de dépendance	**Fait**	**Fait**	**Fait**	
Énumération	**Fait**	**Fait**	**Fait**	
Classe abstraite	**Fait**	**Fait**	**Fait**	
Interface	**Fait**	**Fait**	**Fait**	
Classe	**Fait**	**Fait**	**Fait**	
Note	**Fait**	**Fait**	**Fait**	
Paquetage	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
Diagramme de séquence				
Éléments	Ajouter	Supprimer	Afficher/cacher	
Message	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
Acteur	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
Objet	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
Note	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
for	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
alt/else	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
ref	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
loop	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**
opt	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Annulé (Optionnel)**	**Ann**

Tests de déplacement et de changement de couleur d'entités :
(Ne pas oublier que le déplacement d'une entité reliée à une relation doit faire suivre la relation)

Identifiant :	TEST_19	
Identifiant de l'exigence :	MOD_40 - MOD_50	
Diagramme de cas d'utilisation		
Entité	Déplacer	Modifier la couleur
Note	**Fait**	**Fait**
Frontière	**Annulé (Optionnel)**	**Annulé (Optionnel)**
Acteur	**Fait**	**Fait**
Cas d'utilisation	**Fait**	**Fait**
Diagramme de classe		
Entité	Déplacer	Modifier la couleur
Énumération	**Fait**	**Fait**
Classe abstraite	**Fait**	**Fait**
Interface	**Fait**	**Fait**
Classe	**Fait**	**Fait**
Note	**Fait**	**Fait**
Paquetage	**Annulé (Optionnel)**	**Annulé (Optionnel)**
Diagramme de séquence		
Entité	Déplacer	Modifier la couleur
Acteur	**Annulé (Optionnel)**	**Annulé (Optionnel)**
Objet	**Annulé (Optionnel)**	**Annulé (Optionnel)**
Note	**Annulé (Optionnel)**	**Annulé (Optionnel)**
for	**Annulé (Optionnel)**	**Annulé (Optionnel)**
alt/else	**Annulé (Optionnel)**	**Annulé (Optionnel)**
ref	**Annulé (Optionnel)**	**Annulé (Optionnel)**
loop	**Annulé (Optionnel)**	**Annulé (Optionnel)**
opt	**Annulé (Optionnel)**	**Annulé (Optionnel)**

Identifiant :	TEST_20	
Identifiant de l'exigence :	MOD_70	
Objet :	Modifier le style d'une flèche	
Action :	Modifier le style d'une flèche de toutes les manières possibles	
Résultat attendu :	Le style de la flèche a été changé visuellement et respecte l'UML2	
État :	**Fait**	Le : 03/05/2016

Identifiant :	TEST_21	
Identifiant de l'exigence :	MOD_110	
Objet :	Inverser le sens d'une flèche	
Action :	Inverser le sens d'une flèche de toutes les manières possibles	
Résultat attendu :	Le sens de la flèche a été changé et respecte UML2	
État :	**Fait**	Le : 03/05/2016

Modification d'un diagramme de cas d'utilisation			
Identifiant :	TEST_22		
Numéro	**Description**	**Priorité**	**Testé et approuvé**
USE_10	Modifier le texte d'une note	Indispensable	**Fait**
USE_20	Modifier le nom d'une frontière. Ce nom est une chaîne de caractères quelconque.	Indispensable	**Fait**
USE_30	Modifier le nom d'un acteur. Ce nom est une chaîne de caractères quelconque.	Indispensable	**Fait**
USE_40	Modifier le texte d'un cas d'utilisation. Ce texte est une chaîne de caractères quelconque.	Indispensable	**Fait**

Modification d'un diagramme de classes			
Identifiant :	TEST_23		
Identification	**Description**	**Priorité**	**Testé et approuvé**
CLA_10	Modifier le nom d'un objet	Indispensable	**Fait**
CLA_20	Changer le type d'un objet	Indispensable	**Fait**
CLA_30	Modifier un attribut	Indispensable	**Fait**
CLA_40	Modifier une méthode	Indispensable	**Fait**
CLA_50	Modifier la visibilité d'un attribut ou d'une méthode	Indispensable	**Fait**
CLA_60	Modifier le texte d'une note	Indispensable	**Fait**
CLA_70	Modifier le nom d'un paquetage	Indispensable	**Annulé (Optionnel)**
CLA_80	Modifier un objet (son nom, ses attributs et ses méthodes) contenu par un paquetage	Indispensable	**Fait**
CLA_90	Redimensionner un paquetage et un objet	Indispensable	**Fait**

Modification d'un diagramme de séquence			
Identifiant :	TEST_24		
Numéro	**Description**	**Priorité**	**Testé et approuvé**
SEQ_10	Modifier le nom d'un acteur. Ce nom est une chaîne de caractères quelconque	Indispensable	**Annulé (Optionnel)**
SEQ_20	Modifier le nom d'un objet	Indispensable	**Annulé (Optionnel)**
SEQ_30	Modifier le texte d'une note	Indispensable	**Annulé (Optionnel)**
SEQ_40	Modifier le type d'un cadre d'itération : for, alt, ref, etc.	Indispensable	**Annulé (Optionnel)**
SEQ_50	Modifier le texte contenu dans un cadre d'itération	Indispensable	**Annulé (Optionnel)**
SEQ_60	Modifier les messages contenus dans un cadre d'itération	Indispensable	**Annulé (Optionnel)**
SEQ_70	Modifier la ligne de temps d'un objet	Indispensable	**Annulé (Optionnel)**

7 Test des exigences fontionnelles

Exigences fonctionnelles

L'exigence EXF_10 n'a pas de test associé. L'exigence est triviale.
TEST_16 vérifie EXF_30.
TEST_11 vérifie EXF_50.

Test

L'ergonomie

Identifiant :	TEST_25	
Identifiant de l'exigence :	EXF_20	
Objet :	L'ergonomie	
Action :	Réalisation d'un sondage pour tester la facilité et l'efficacité de l'application	
Résultat attendu :	Toutes les actions doivent être faciles d'utilisation. On doit pouvoir faire n'importe quelle action en au plus 4 clics. Manipuler un diagramme doit être facile.	
Test supplémentaire :	On pourra faire tester les versions graphiques au client, mais aussi aux étudiants sans leur expliquer comment fonctionne l'application. S'ils arrivent facilement et rapidement à maîtriser le logiciel alors le test sera réussi. On prendra également leurs avis et leurs conseils.	
État :	**Fait**	Le : 03/05/2016

UML2

Identifiant :	TEST_26	
Identifiant de l'exigence :	EXF_40	
Objet :	UML2	
Action :	Manipuler les diagrammes	
Résultat attendu :	Toutes les actions possibles en UML2 doivent pouvoir être faites avec l'application	
Test supplémentaire :	Lister toutes les actions possibles en UML2 avec le client et les effectuer sur le produit.	
État :	**Fait**	Le : 03/05/2016

Entités de même nom

Identifiant :	TEST_27	
Identifiant de l'exigence :	EXF_60	
Objet :	Entités de même nom	
Action :	Nommer deux entités dans un même diagramme avec le même nom	
Résultat attendu :	L'application prévient que ce n'est pas possible	
État :	**Fait**	Le : 03/05/2016

Modulaire

Identifiant :	TEST_28	
Identifiant de l'exigence :	EXF_70	
Objet :	Modulaire	
Action :	Une fois la structure logicielle terminée, on doit pouvoir rajouter des fonctionnalités facilement en étendant le code	
Résultat attendu :	L'extension du code doit être possible. Imaginer à la fin de la rédaction du DAL la façon dont nous pouvons étendre l'application pour implémenter des fonctionnalités telles que l'édition de nouveaux diagrammes.	
État :	**Fait**	Le : 03/05/2016

Fonctionnel sur les ordinateurs de l'université

Identifiant :	TEST_28	
Identifiant de l'exigence :	EXF_80	
Objet :	Fonctionnel sur les ordinateur de l'université	
Action :	Tester l'application à l'université	
Résultat attendu :	L'application doit parfaitement marcher	
État :	**Fait**	Le : 03/05/2016

8 Qualité

Toutes les fonctionnalités doivent être faites rapidement sur l'application (moins d'une seconde pour chaque action). Le reverse devra s'exécuter en moins de 30 secondes.

9 Satisfaction du client

Le but final de l'application est de satisfaire le client. Pour ce faire, les tests ici présents vérifient que l'application finale respecte bien tous les points de la STB. Chaque cas d'utilisation doit être implanté. Les tests ici présents doivent également tous être respectés. On veillera à ce qu'il ne reste plus de « **Fait** ». De plus, on effectuera des livraisons régulières au client (tous les mois) en s'assurant qu'il est parfaitement satisfait de ce qu'on a fait. On mettra l'accent sur l'ergonomie de l'application et sur la qualité.