Spécification Technique des Besoins
Projet UML Reverse

20 mai 2016

Version :	0.1
Date :	20 mai 2016
Rédigé par :	Nabil Belkhous Stephen Cauchois Anthony Godin Yohann Henry Florian Inchingolo Nicolas Meniel Saad Mrabet
Relu par :	Florian Inchingolo Nicolas Meniel Guillaume Leroy
Approuvé par :	Stéphane Hérauville
Signature :	

Mises à jour

Version	Date	Modifications réalisées
0.1	29/12/2015	Fin de la rédaction de a STB
0.1	11/11/2015	Début de la rédaction de la STB

Table des matières

1 Objet du projet

Le projet **UML Reverse** vise à développer un logiciel permettant de construire des diagrammes UML graphiquement. Le projet est décomposé en deux parties :

1.1 Reverse

Le programme produit un diagramme UML à partir d'un code source. Pour cette version, seul le code Java est concerné.
Les diagrammes créés seront sauvegardés en PlantUML et pourront être modifiés grâce au générateur de diagramme.
Le reverse engineering ne gérera pas les cardinalités et pourra être executé au moins en moins de 30 secondes.

1.2 Générateur de diagramme

Le générateur de diagramme permet à l'utilisateur de construire son propre diagramme UML graphiquement.
Il pourra soit en créer un nouveau, soit en charger un à partir d'un fichier PlantUML (avec éventuellement un fichier de paramètres propre au logiciel).
Le générateur permettra de modifier tous les éléments des diagrammes ainsi que leurs positions en *drag and drop* (glisser-déposer).
L'utilisateur pourra sauvegarder ses diagrammes avec ses paramètres. ou bien les exporter.

2 Documents applicables de référence

2.1 Expression du besoin du projet UML Reverse

L'utilisation des diagrammes UML fait partie des outils de tout développeur informatique. Les applications (ou plugins) de bon niveau permettant d'utiliser efficacement ces diagrammes sont limités dans le domaine du libre, et ne disposent pas de toutes les fonctionnalités utiles.
Remarque : Les logiciels disponibles en libre ne sont pas pérennes. En effet, souvent rachetés par des entreprises, les allers-retours entre versions libres et propriétaires sont très fréquents. Quant aux versions restées libres, elles ont pour défaut leur manque de fonctionnalités, l'ergonomie très discutable et le manque de documentation fiable.

Objectifs

Développer une application ergonomique permettant d'effectuer rapidement un développement informatique intégrant des diagrammes UML.

Objectifs imposés

Afin de limiter le périmètre de l'application et d'en faire un outil exploitable, les contraintes imposées sont les suivantes :

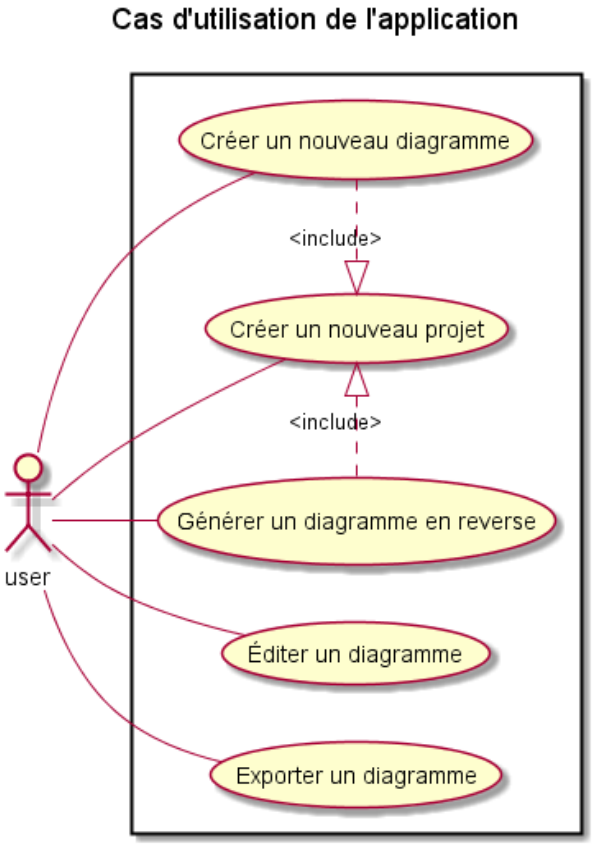
- L'application devra respecter le pseudo-langage PlantUML pour la définition des schémas. Des informations complémentaires devront être générées pour les éléments non pris en compte par PlantUML (paramètres graphiques, etc.)
- L'application doit permettre de créer, modifier et présenter graphiquement les diagrammes dans le respect d'UML 2. Une interface graphique doit permettre de définir les paramètres graphiques :
  - positionnement des éléments ;
  - sélection des éléments affichés/cachés (exemple : dans le diagramme de classe, noms de méthodes seuls ou détails).
- L'application doit permettre le *reverse engineering*, c'est-à-dire l'extraction du diagramme de classe à partir du code source en Java d'une application. Le résultat sera stocké au format PlantUML.

3 Lexique

Se référer au doccument Terminologie.

4 Fonctionnalités

4.1 Diagramme de cas d'utilisation



4.2 Exigences fonctionnelles de l'IHM

Cas d'utilisation de l'IHM		
Identification	**Description**	**Priorité**
IHM_10	Créer un nouveau projet. (L'application crée une arboresence de dossiers pour y enregistrer par la suite des diagrammes)	Indispensable
IHM_20	Éditer un projet	Indispensable
IHM_30	Éditer un diagramme (suppose l'ouverture, la modification,la sauvegarde et la supression) spécifique à notre application avec ses paramètres	Indispensable
IHM_40	Modifier le zoom du diagramme	Indispensable
IHM_50	Charger un fichier de paramètres pour tous les diagrammes	Indispensable
IHM_60	Expoter un fichier de paramètres qui va contenir le style	Indispensable
IHM_70	Exporter un diagramme au format PlantUML	Indispensable
IHM_80	Exporter un diagramme au format image/PDF	Optionnelle
IHM_90	Imprimer un diagramme	Important
IHM_100	Importer/charger un diagramme UML écrit en PlantUML compatible dans un projet	Indispensable

4.3 Exigences fonctionnelles des générations de diagrammes

Cas d'utilisation des diagrammes		
Identification	**Description**	**Priorité**
DIA_10	Créer un nouveau diagramme de cas d'utilisation	Indispensable
DIA_20	Créer un nouveau diagramme de classes	Indispensable
DIA_30	Créer un nouveau diagramme de séquence	Optionnelle
DIA_40	Créer un nouveau diagramme de paquetages	Optionnelle
DIA_50	Créer un nouveau diagramme d'états	Optionnelle
DIA_60	Générer un diagramme de classes par *reverse engineering* sur du code Java	Indispensable
DIA_70	Générer un diagramme de paquetages par *reverse engineering* sur du code Java	Optionnelle
DIA_80	Générer un diagramme de séquence par *reverse engineering* sur du code Java	Optionnelle

4.4 Exigences fonctionnelles des modifications possibles

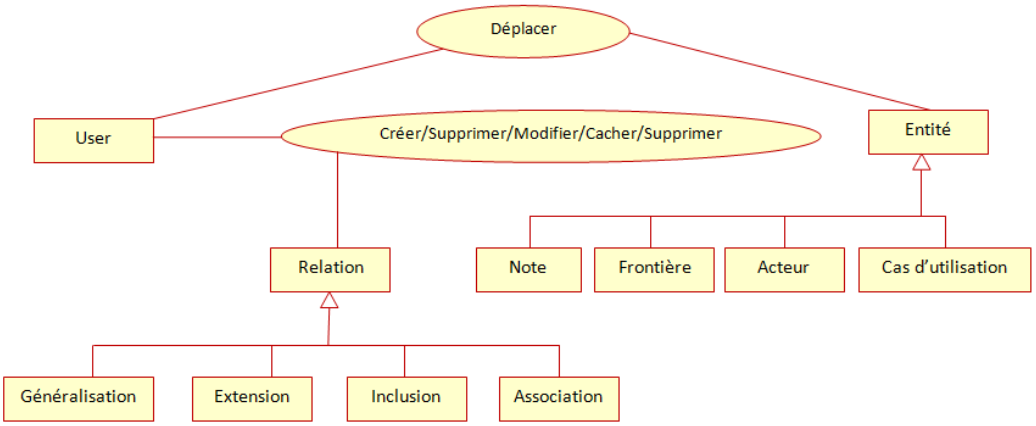
Cas d'utilisation des modifications possibles		
Identification	**Description**	**Priorité**
MOD_10	Ajouter un élément	Indispensable
MOD_20	Supprimer un élément	Indispensable
MOD_30	Afficher/cacher un élément	Indispensable
MOD_40	Déplacer une entité	Indispensable
MOD_50	Modifier la couleur d'une entité	Optionnelle
MOD_60	Ajouter une flèche entre deux entités	Indispensable
MOD_70	Modifier le style d'une flèche	Indispensable
MOD_80	Modifier un des champs d'une flèche	Indispensable
MOD_90	Afficher/cacher une flèche	Indispensable
MOD_100	Supprimer une flèche	Indispensable
MOD_110	Inverser le sens d'une flèche	Optionnelle

4.5 Précisions

— Déplacer une entité déplace tout ce qui lui est lié.
— Cacher/afficher une entité cache/affiche tout ce qui lui est relié.
— Éditer un diagramme exige qu'un projet soit déjà créé.
— Tous les diagrammes peuvent contenir un titre (texte). Il peut être modifié.

4.6 Modification d'un diagramme de cas d'utilisation

4.6.1 Schema fonctionnel d'utilisation



4.6.2 Précisions

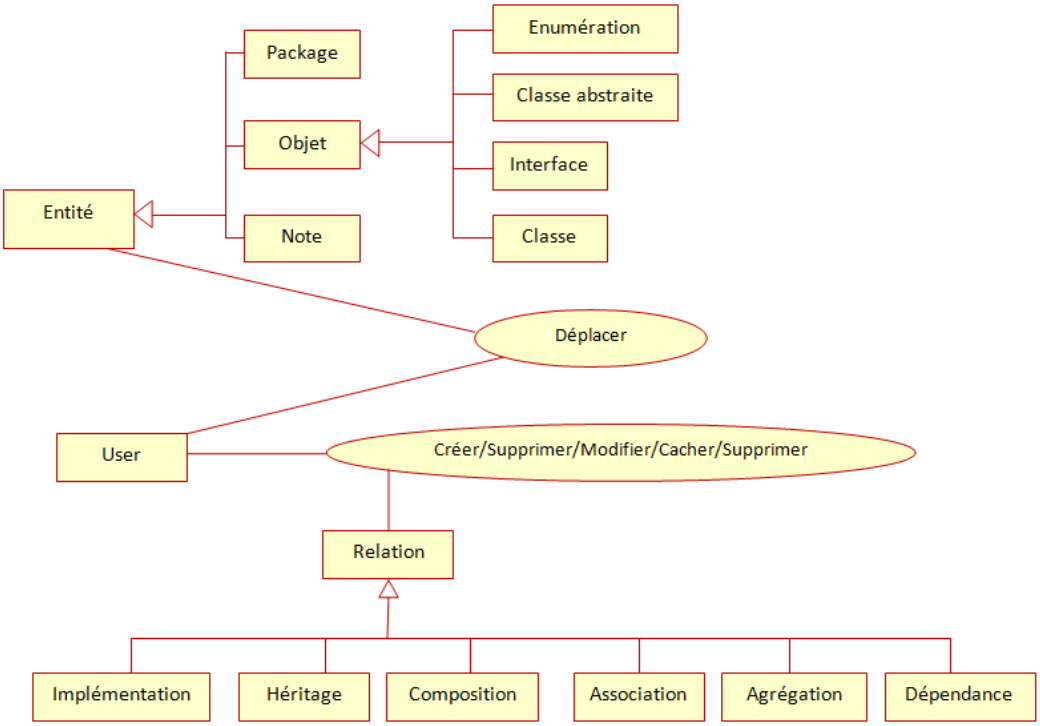
- Une note contient un texte quelconque.
- Une frontière contient un nom.
- Un acteur contient un nom.
- Un cas d'utilisation contient un texte. Ce texte peut contenir des séparations en trait continu, en pointillés ou en double trait continu.
- Une relation est une flèche.

4.6.3 Liste des cas d'utilisation supplémentaires

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

4.7 Modification d'un diagramme de classes

4.7.1 Schéma fonctionnel d'utilisation



4.7.2 Précisions

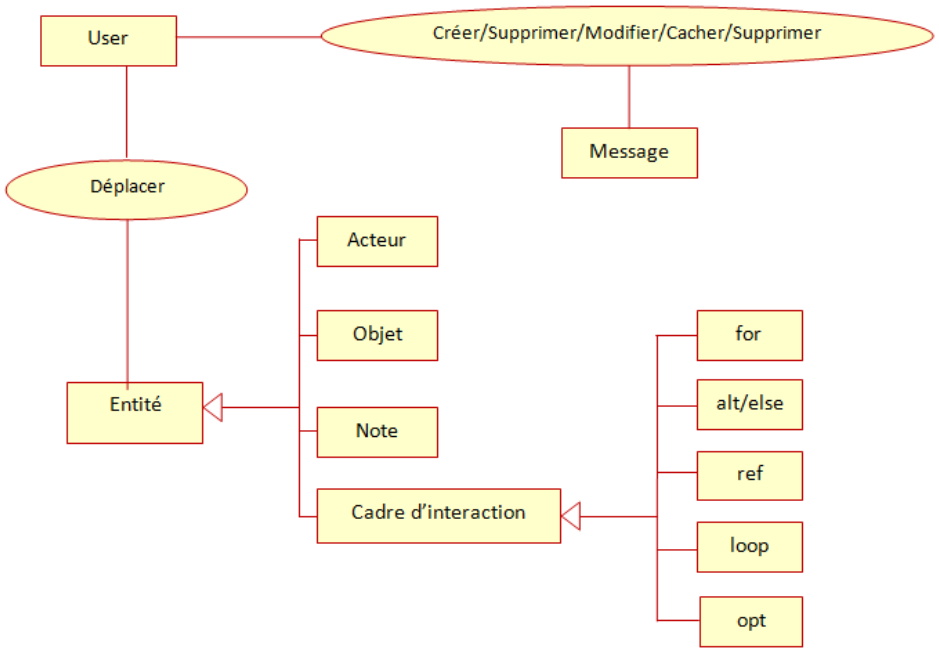
- Les objets contiennent tous les éléments suivants :
  - au moins un nom : un mot seulement ;
  - au moins un type : classe, interface, etc. ;
  - éventuellement des attributs : un texte quelconque sans parenthèse ouvrante ni fermante ;
  - éventuellement des méthodes : un texte quelconque contenant au moins une parenthèse ouvrante ou fermante.
- Un attribut et une méthode peuvent avoir une (et une seule) visibilité éditable : publique, privée, protégée ou *package-private*.
- Une note contient un texte.
- Une relation est une flèche.
- Un paquetage contient un nom et éventuellement des objets qui ne sont pas considérés comme des entités mais comme des éléments.

4.7.3 Liste des cas d'utilisation supplémentaires

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

4.8 Cas d'utilisation de modification d'un diagramme de séquence

4.8.1 Schéma fonctionnel d'utilisation



4.8.2 Précisions

- Un acteur contient un nom.
- Un objet contient un nom (un unique mot) et d'une ligne de « temps de vie ».
- Une note contient un texte.
- Une relation est une flèche.
- Un cadre d'itération contient un texte et regroupe des messages.
- Un cadre else suit imédiatement un cadre alt. Sans alt, il ne peut y avoir de else.
- Un message est une flèche.
- Déplacer un objet déplace aussi tout ce qui lui est lié, c'est-à-dire les flèches, les cadres d'itération et sa ligne de vie.

4.8.3 Liste des cas d'utilisation supplémentaires

4.8.4 Liste des cas d'utilisation

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

5 Exigences fonctionnelles

Identifiant	**Description**
EXF_10	L'application est codée en Java.
EXF_20	L'interface graphique doit être fonctionnelle, pratique et adaptée aux besoins.
EXF_30	Le *reverse* permet de générer un diagramme à partir d'un fichier compatible Java 7.
EXF_40	L'application implémente tous les élements d'un diagramme UML2 pour les diagrammes compatibles.
EXF_50	À tout moment de l'édition d'un diagramme non vide, un fichier PlantUML compilable peut être généré à partir du diagramme.
EXF_60	L'utilisateur ne peut pas avoir deux entités du même nom dans un diagramme.
EXF_70	Le code de l'application doit être modulaire.
EXF_80	L'application doit fonctionner sur les ordinateurs de l'Université.

6 Exigences

6.1 Exigences fonctionnelles

Identifiant	**Description**
EXF_20	L'interface graphique doit être fonctionnelle, pratique et adaptée aux besoins.
EXF_30	Le *reverse* permet de générer un diagramme à partir d'un fichier compatible Java 7.
EXF_40	L'application implémente tous les élements d'un diagramme UML2 pour les diagrammes compatibles.
EXF_50	À tout moment de l'édition d'un diagramme non vide, un fichier PlantUML compilable peut être généré à partir du diagramme.
EXF_60	L'utilisateur ne peut pas avoir deux entités du même nom dans un diagramme.

6.2 Exigences de qualités

Identifiant	**Description**
EXF_70	Le code de l'application doit être modulaire.

6.3 Exigences techniques

Identifiant	Description
EXF_10	L'application est codée en Java.
EXF_80	L'application doit fonctionner sur les ordinateurs de l'Université.