RAPPORT GÉNÉRAL
Itération 1

[redacted] Édouard Lumet

[redacted]

5 juin 2018

1 Rappel du sujet

Notre projet consiste en la création d'une application avec interface graphique ayant pour utilité de :

- visualiser des diagrammes de classe UML,
- générer un diagramme de classe UML à partir du code JAVA, et inversement,
- éditer un diagramme ou du code JAVA.

2 Fonctionnalités et User Stories

Les principales fonctionnalités sont la visualisation, la génération, l'édition et l'exportation.
Les User Stories effectuées lors de cette première itération sont les suivantes :

- « En tant qu'utilisateur je souhaite visualiser le diagramme de classe et le diagramme d'analyse afin d'avoir une représentation conceptuelle et visualiser l'implantation du code. » (40 PT)
- « En tant qu'utilisateur je souhaite à partir du code JAVA, générer un diagramme de classes afin de faciliter la visualisation du code. » (40 PT)

3 Découpage de l'application

Le programme suit une architecture en deux sous-systèmes majeurs :

- Un sous-système d'édition de diagramme, qui contient lui-même trois sous-système :
  - modèle : la représentation en Java d'un diagramme UML et de ces éléments,
  - vue : La représentation graphique de l'éditeur,
  - contrôleur : Le code java permettant à l'utilisateur de manipuler la vue
- Un sous-système dédié à la génération de code

4 Diagrammes de classe

Les diagrammes de classe modélisant notre projet, notre application sont multiples car nous avons un diagramme par fonctionnalité.

En annexe 1 figure le diagramme modélisant la partie modèle du sous-système éditeur.

En annexe 2 figure le diagramme modélisant le sous-système de génération.

5 Choix de conception, problèmes et solutions

Les principaux choix de conception et réalisation ont été tout d'abord d'appliquer un patron d'architecture qui est MVC (modèle-vue-contrôleur).

Le patron Visiteur a été pensé pour la mise en place des fonctionnalités futures traitant de la génération de code.

Lors du développement divers problèmes ont été soulevés en rapport avec la visualisation des éléments. Tout d'abord le framework JavaFX a été proposé comme base graphique pour ses nombreux composants qui pouvait nous faciliter le développement mais ce dernier étant très récent et par manque d'expérience au sein de l'équipe, le choix s'est porté sur Swing afin d'éviter toute perte de temps.

La représentation des éléments est un second problème, entre autre l'organisation des relations dans le plan (comment savoir si deux relations ne se recoupent pas ? comment faire pour qu'une relation ne passe pas sur une entité ?).

6 Organisation et méthode agile

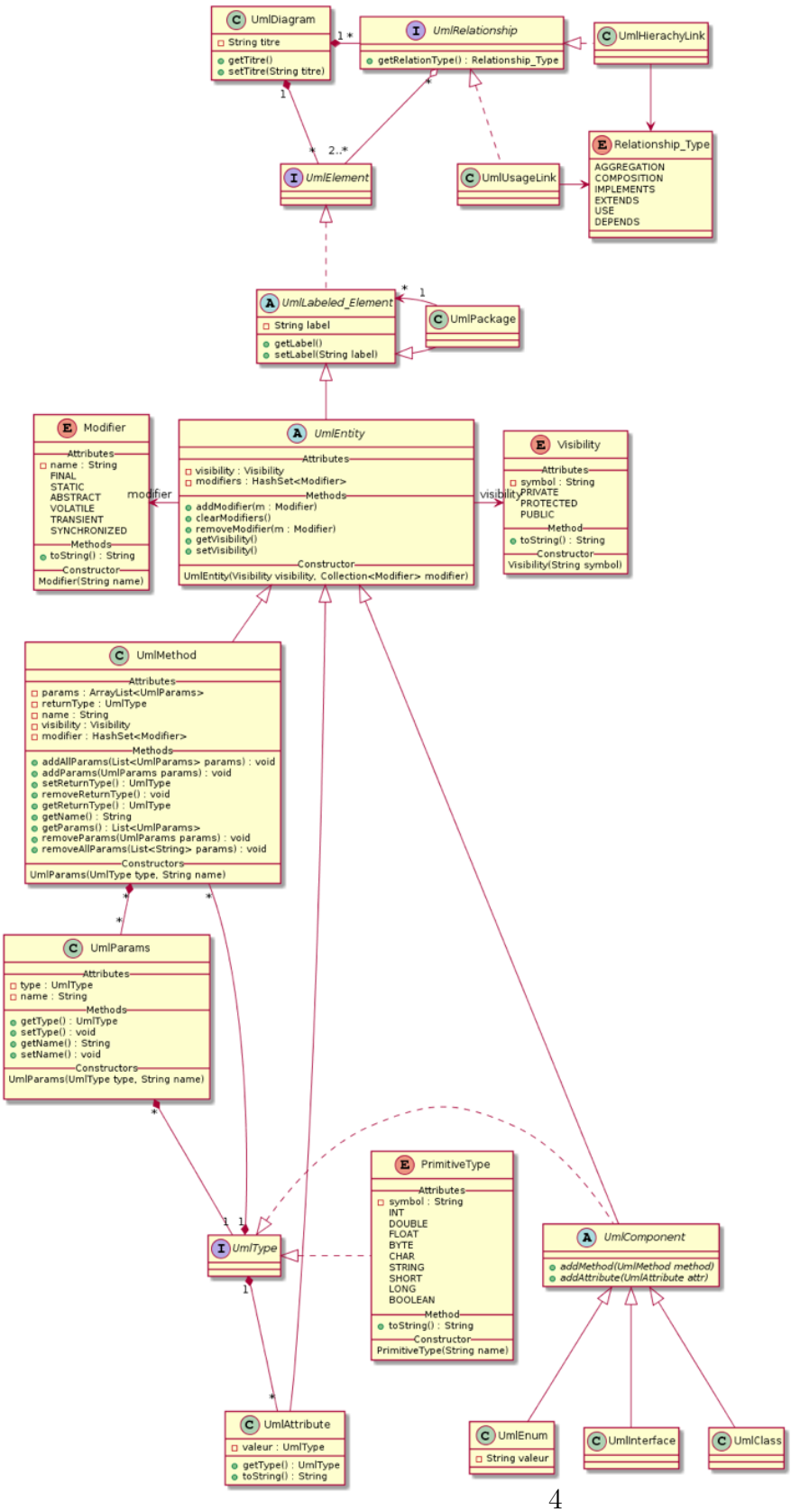
L'organisation consiste en une répartition individuelle des tâches techniques entre les membres du groupe, tout en utilisant Github pour la gestion de sources et WeKan pour la gestion de projet (backlog, US, ...).

La mise en oeuvre des méthodes agiles a conduit dans un premier temps à l'écriture de User Stories et de Technical Tasks ainsi qu'à une prévision du travail effectué lors du Sprint 1.

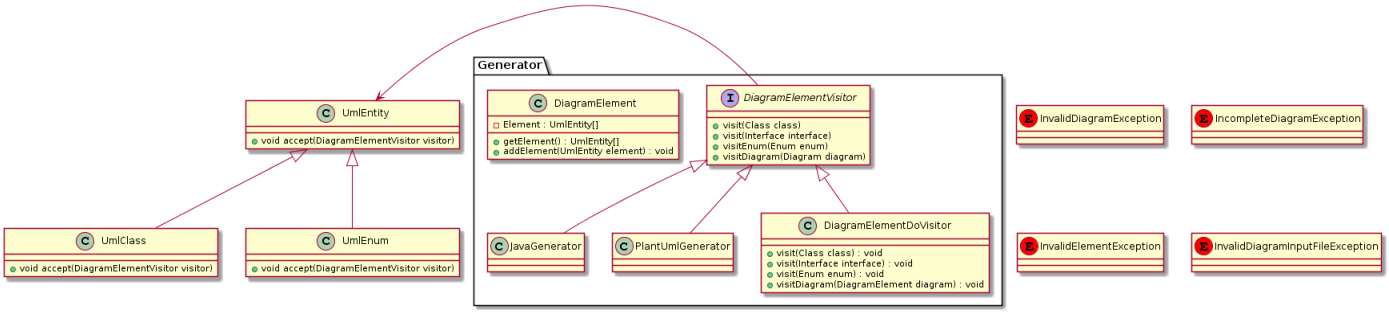
Cependant cette première mise en oeuvre s'est avérée être un échec, l'ensemble du groupe étant parti directement sur une conception de l'ensemble de la solution avec la création d'un diagramme de classe UML. Ce problème ne nous a pas permis de terminer l'ensemble des tâches dans les US.

Après en avoir pris conscience au cours du Sprint 1, le backlog a été retravaillé et les User Stories re-validées afin de préparer au mieux le Sprint 2. Le diagramme produit n'a pas été utilisé tel quel mais a permis d'avoir une idée plus claire de l'architecture générale du programme.

A Annexe 1


UmlDiagram
String titre
getTitre()
setTitre(String titre)
UmlRelationship
getRelationType() : Relationship_Type
UmlHierachyLink
Relationship_Type
AGGREGATION
COMPOSITION
IMPLEMENTS
EXTENDS
USE
DEPENDS
UmlElement
UmlUsageLink
UmlLabeled_Element
String label
getLabel()
setLabel(String label)
UmlPackage
Modifier
Attributes
name : String
FINAL
STATIC
ABSTRACT
VOLATILE
TRANSIENT
SYNCHRONIZED
Methods
toString() : String
Constructor
Modifier(String name)
modifier
UmlEntity
Attributes
visibility : Visibility
modifiers : HashSet<Modifier>
Methods
addModifier(m : Modifier)
clearModifiers()
removeModifier(m : Modifier)
getVisibility()
setVisibility()
Constructor
UmlEntity(Visibility visibility, Collection<Modifier> modifier)
visibility
Visibility
Attributes
symbol : String
PRIVATE
PROTECTED
PUBLIC
Method
toString() : String
Constructor
Visibility(String symbol)
UmlMethod
Attributes
params : ArrayList<UmlParams>
returnType : UmlType
name : String
visibility : Visibility
modifier : HashSet<Modifier>
Methods
addAllParams(List<UmlParams> params) : void
addParams(UmlParams params) : void
setReturnType() : UmlType
removeReturnType() : void
getReturnType() : UmlType
getName() : String
getParams() : List<UmlParams>
removeParams(UmlParams params) : void
removeAllParams(List<String> params) : void
Constructors
UmlParams(UmlType type, String name)
UmlParams
Attributes
type : UmlType
name : String
Methods
getType() : UmlType
setType() : void
getName() : String
setName() : void
Constructors
UmlParams(UmlType type, String name)
PrimitiveType
Attributes
symbol : String
INT
DOUBLE
FLOAT
BYTE
CHAR
STRING
SHORT
LONG
BOOLEAN
Method
toString() : String
Constructor
PrimitiveType(String name)
UmlType
UmlComponent
addMethod(UmlMethod method)
addAttribute(UmlAttribute attr)
UmlAttribute
valeur : UmlType
getType() : UmlType
toString() : String
UmlEnum
String valeur
UmlInterface
UmlClass

B Annexe 2


Generator
UmlEntity
void accept(DiagramElementVisitor visitor)
DiagramElement
Element : UmlEntity[]
getElement() : UmlEntity[]
addElement(UmlEntity element) : void
DiagramElementVisitor
visit(Class class)
visit(Interface interface)
visitEnum(Enum enum)
visitDiagram(Diagram diagram)
InvalidDiagramException
IncompleteDiagramException
UmlClass
void accept(DiagramElementVisitor visitor)
UmlEnum
void accept(DiagramElementVisitor visitor)
JavaGenerator
PlantUmlGenerator
DiagramElementDoVisitor
visit(Class class) : void
visit(Interface interface) : void
visit(Enum enum) : void
visitDiagram(DiagramElement diagram) : void
InvalidElementException
InvalidDiagramInputFileException