Université de Montréal

Rapport Travail Pratique #2

Par

Charbel Kassis, p0976458

Département d’informatique et de recherché opérationnelle

Faculté des arts et sciences

Travail présenté à Edouard BATOT et Lechanceux LUHUNU KAVUYA dans le cadre du cours IFT3913,

Qualité du logiciel et métriques

2 Novembre 2016

# Les nouveau packages :

uml.metrique : ce package contient les classes qui représentent des métriques, ainsi que la classe MetriqueBuilder pour construire des objets métriques.

# Les changements au anciens packages et classes:

Package uml.gui.controller

* uml.gui.controller.UML\_GUI\_Controller : Ajout du “look and feel” du system d’exploitation

en cours d’utilisation pour les fenêtres Swing. Ajout des nouveaux listeners liés aux métriques.

* uml.gui.controller.UML\_ClassListener : Ajout d’une fonctionnalité qui efface la liste des métriques courantes lorsqu’on choisit une autre classe.
* uml.gui.controller.UML\_MetriqueListener : Nouvelle classe. Un listener pour le bouton qui déclenchera l’affichage des métriques.
* uml.gui.controller.UML\_MetriqueMouseListener : Nouvelle classe. Un listener qui écoute les cliques sur les métriques pour afficher leur définition dans une fenêtre ainsi que d’afficher immédiatement les définitions sous forme de tooltips (mettre le délai à 0 seconde).
* uml.gui.controller.UML\_MetriqueTooltipListener : Nouvelle classe. Un listener pour afficher les tooltips des métriques.
* uml.gui.controller.UML\_CSV\_MetriqueListener : Nouvelle classe. Un listener pour le bouton qui sert à générer un fichier CSV des métriques.

Package uml.gui

* uml.gui.UML\_FileChooser : Ajout des deux boutons pour afficher les métriques et pour générer un fichier CSV. Ajout de tooltips sur tous les boutons se trouvant dans UML\_FileChooser.
* uml.gui.UML\_List : Ajout d’un générique pour la JList. JList<String> dans la version du tp1, JList<T> dans la version du tp2, où T est String dans tous les cas sauf pour UML\_MetriqueList, où on utilise une JList<AbstractMetrique>, la JList affichera le toString de l’objet AbstractMetrique (voir plus bas)
* uml.gui.UML\_MetriqueList : Nouvelle classe. Classe qui représente la fenêtre des métriques.

Package uml.metrique (nouveau package)

* uml.metrique.AbstractMetrique : Nouvelle classe. Classe abstraite qui représente une métrique. Il faut définir une méthode pour calculer la métrique, pour retourner les tooltips et retourner le nom de la métrique dans ses sous-classes non abstraites. le toString d’une AbstractMetrique renvoi [nom de métrique] : [valeur de métrique], c’est-à-dire la valeur textuelle qui sera afficher dans la liste des métriques.
* uml.metrique.MetriqueBuilder : Nouvelle classe. Classe qui sert à construire des métriques. Vu que les métriques vont utiliser le même nom de la classe sélectionnée ainsi que la même liste de Généralisation, Agrégation, etc. Utiliser un builder pour les construire rendrait le code plus simple à lire, car il suffit de passer tous ces paramètres au builder pour qu’il leurs assigne ces paramètres un après les autres.
* uml.metrique.XYZ\_Metrique : Nouvelles classes. XYZ est le nom de chaque métrique à calculer (ANA\_Metrique, CAC\_Metrique, CLD\_Metrique, etc.) qui sous-classe AbstractMetrique.

Package uml.struct

* uml.struct.Tree : Nouvelle classe. Une classe qui représente un arbre. Elle est utilisée pour faire des calculs pour les métriques en relation avec les sous-classes et superclasses en construisant l’arbre d’héritage et en le parcourant pour faire des calculs (le nombre d’enfants et la profondeur).

# Utilisation :

La partie traitée dans le TP1 fonctionne toujours de la même façon : On clique sur « Charger fichier », on choisit le fichier à parser puis la liste des classes apparaît si le parsing est réussi. Dans le tp2, deux boutons et un panneau ont été ajoutées. Le bouton « Générer fichier CSV » ouvrira un navigateur de fichier pour choisir l’emplacement où on veut enregistrer le fichier CSV. Si aucun fichier n’a été parsé, cliquer sur « Générer fichier CSV » ouvrira une boîte de dialogue pour afficher un message d’erreur. Lorsque le fichier a été généré sans problème, une boîte de dialogue apparaîtra pour afficher deux messages : l’opération a été terminée et le lien vers le fichier. Le bouton « Calculer métriques » affichera les métriques dans la liste des métriques. Si aucune classe n’a été sélectionnée ou qu’aucun fichier n’a été parsé, une boîte de dialogue s’affichera pour dire qu’aucune classe n’est sélectionnée. Il y a deux façon pour voir les définitions des métriques : 1) mettre la souris sur l’élément dont on veut voir la définition, un tooltip apparaitra avec la définition. 2) cliquer sur l’élément de la liste, une boîte de dialogue apparaitra avec la définition.

# Déroulement :

La partie traitée dans le TP1 se déroule toujours de la même façon. Les nouveaux changements sont pour les deux nouveaux boutons :

lorsqu’on clique sur « Afficher métriques », le UM\_MetriqueListener appelera le showList() du panneau UML\_MetriqueList en lui passant tout le modèle dans setList(). UML\_Metrique liste va ensuite générer des map pour les Généralisations/Agrégation/etc pour les passer par la suite au builder de Métrique :

/\* instancier le MetriqueBuilder puis lui ajouter les listes necessaires pour construire les objets AbstractMetrique \*/

MetriqueBuilder builder =

**new** MetriqueBuilder().setClassMap(**this**.umlClassesMap)

.setGeneralizationMap(**this**.umlGeneralizationMap)

.setAggregationList(**this**.model.getAggregations())

.setAssociationlist(**this**.model.getAssociations())

.setSelectedClassName(**this**.selectedClassName);

/\* construire les metriques puis ajouter eur texte dans l'objet DefaultListModel pour qu'il soit afficher

\* dans la liste des metriques \*/

listModel.addElement((T) builder.build(***ANA***));

listModel.addElement((T) builder.build(***NOM***));

...

}

Ainsi, les métriques seront affichées en utilisant le toString de chaque objet de type AbstractMetrique. Chaque métrique à sa propre implémentation de calculateMetrique et getMetriqueName :

**public** String toString() {

**return** **this**.getMetriqueName()+" : "+**this**.calculateMetrique();

}

Lorsqu’on clique sur « Générer fichier CSV », le UML\_CSV\_MetriqueListener créera une JFileChooser pour choisir un répertoire où générer le fichier [nom du modèle].csv

File directory = fileChooser.getSelectedFile();

String fileName = **this**.parser.getModel().getIdentifier() + ".csv";

/\* le path complet du fichier csv est: (repertoire choisi par utilisateur)/(nom du model).csv \*/

File csvFile = **new** File(directory,fileName);

Par la suite, un PrintWriter sera créé sur le fichier csvFile pour ouvrir un stream d’écriture. On va, comme pour l’affichage des métriques individuelles, se servir du MetriqueBuilder pour construire les métriques, mais vu qu’on se s’intéresse qu’à la valeur, on va utiliser calculateMetrique() directement.

Voici la méthode qui génère une ligne (row) du fichier csv.

String row = "";

MetriqueBuilder builder = **...**

/\* chercher toutes les metriques, de ANA jusqu'a NOD \*/

**for**(**int** i=***ANA*** ; i<=***NOD*** ; i++) {

/\* ajouter la valuer de la metrique \*/

row += builder.build(i).calculateMetrique();

**if**(i!= ***NOD***)

row += ",";

} **return** row;

A la fin de l’écriture, si tout s’est bien déroulé, un message apparaîtra.

JOptionPane.*showMessageDialog*(**this**.frame, "Le fichier "+fileName+" a été créé."

+ "\nLien du fichier: "+csvFile.getAbsolutePath());

# C:\Users\Cher\Dropbox\IFT3913\tp2\uml uml.pngDiagramme UML :

Ce diagramme est une vue d’ensemble générale du programme. Il lui manque les sous-classes des classes abstraites et les exceptions (sous-classes de Extractor, AbstractUML, AbstractMetrique, UML\_List, les listeners du package uml.gui.controller et les exceptions du package uml.exception) à cause du manque d’espace.

* uml.exception : UML\_ArugmentNotExtractedException, UML\_ClassesNotExtractedException, UML\_Exception, UML\_FileNotParsedException, UML\_IllegalSyntaxErrorException, UML\_MissingSyntaxException, UML\_RolesNotExtractedException, UML\_SyntaxLengthException.
* uml.extraction : ExtractAggregation, ExtractArgument, ExtractAssociation, ExtractAttribute, ExtractClass, ExtractGeneralization, ExtractListDec, ExtractModel, ExtractMultipleRoles, ExtractOperation, ExtractRole, ExtractSubclass.
* uml.gui : UML\_associationAggregationList, UML\_AttributeList, UML\_ClassList, UML\_Details, UML\_FileChooser, UML\_MethodList, UML\_MetriqueList, UML\_SousclasseList.
* uml.gui.controller : UML\_AssocAggreListListener, UML\_ClassListener, UML\_CSV’MetriqueListener, UML\_FileChooserListener, UML\_MetriqueListener, UML\_MetriqueMouseListener, UML\_MetriqueTooltipListener
* uml.metrique : ANA\_Metrique, CAC\_Metrique, CLD\_Metrique, DescendantMetrique, DIT\_Metrique, ETC\_Metrique, ITC\_Metrique, NOA\_Metrique, NOC\_Metrique, NOD\_Metrique, NOM\_Metrique
* uml.struct : UML\_Aggregation, UML\_Association, UML\_Class, UML\_DataItem, UML\_Generalization, UML\_Model, UML\_Operation, UML\_Role.